On the (Im)possibility of Asynchronous Proactive Secret Sharing

Erica Blum[1], Yingzi Gao[2], Julian Loss[3], Yuan Lu[4], Zhenliang Lu[5], and Qiang Tang[6]

[1] Reed College ericablum@reed.edu
[2] Shandong University gaoyingzi@sdu.edu.cn
[3] Ruhr University Bochum julian.loss@rub.de
[4] Agency for Science, Technology and Research lu_yuan@a-star.edu.sg
[5] City University of Hong Kong zhenliang.lu@cityu.edu.hk
[6] The University of Sydney qiang.tang@sydney.edu.au

Abstract. In this paper, we systematically study the (im)possibilities of proactive secret sharing (PSS) in fully asynchronous authenticated networks. To reflect the subtleties arising from the absence of a global clock, we first formalize a hierarchy of natural mobile adversary models for asynchronous PSS (APSS), capturing different adversarial capabilities over corrupted parties and distinct ways of accounting for mobile corruption in asynchrony. Under these models, we then (i) establish new impossibility results by exhibiting generic attacks on APSS, thereby deriving the corresponding resilience upper bounds, and (ii) demonstrate the tightness of these bounds via explicit APSS protocols achieving optimal resilience in each model.

Keywords: Proactive secret sharing · Mobile adversary · Asynchronous network

1 Introduction

(Verifiable) secret sharing [45,27,43] is a foundational primitive widely used in threshold cryptography [24] and secure multiparty computation [10], allowing a designated dealer to share a secret among n parties, such that the secret remains private as long as the adversary corrupts no more than f out of the n parties. This corruption model is fine for short-lived secrets, but there are many applications where secret shares have a lifetime of weeks or months (such as MPC wallets and confidential documents kept on blockchains for conditional release) [18,17,36,12]. In these settings, it is natural to consider an adversary that can take advantage of the secret's longer lifespan to gradually compromise additional parties and accumulate secret shares over time, potentially collecting more than f shares.

Proactive secret sharing (PSS) and mobile adversaries. *Proactive* secret sharing (PSS) [40,33] was introduced to address exactly this threat. Considering that secret shares might be leaked over time, PSS introduces an additional Reshare protocol that enables the participants to periodically "refresh" their shares, i.e., to obtain a new sharing of the same secret. Informally, PSS schemes achieve security against a *mobile* adversary who can release currently corrupted parties in order to corrupt new parties, so long as it never corrupts more than f parties within a single period (called an *epoch*).

Open problems on (in)feasibility of fully asynchronous PSS. While proactive secret sharing (PSS) has been extensively studied in the *synchronous* setting [7,26,33,38,3,39,25,8,4], Asynchronous Proactive Secret Sharing (APSS) schemes have remained underexplored. A fully asynchronous network presents a barrier that is not present in the synchronous setting, and that may initially seem insurmountable: in an asynchronous network, the adversary can prevent participants from coordinating a precise and unanimous moment to execute the Reshare protocol associated with a given "epoch transition". Consequently, the adversary can postpone messages from prior resharings and deliver them during the current resharing, thereby leveraging its current-epoch corruption budget to gain extra advantage from these delayed messages.

Alexandru et al. [2] first pointed out this subtle threat of cross-epoch message interleaving under asynchrony, observing that all existing "APSS" constructions [48,14,31,47] rely on

certain forms of synchrony assumptions. These assumptions, whether implicit or explicit, prevent an asynchronous adversary from interleaving messages across multiple Reshare protocols. The work of Alexandru et al. (ABK+22) then showed that in a fully asynchronous network, such a cross-epoch attack renders a certain form of APSS impossible. Crucially, however, the ABK+22 [2] impossibility analysis is specific to APSS protocols in which the underlying secret sharing has a reconstruction threshold t that coincides with the maximal resilience f. This feature is not inherent to general APSS, and indeed there exist natural "APSS" implementations with $t = 2f > f$ [28,48].

In short, despite several (recent) tentative constructions [48,28,14,31,34,47,6,46] and an established impossibility result for a specific form of APSS [2], the precise boundaries of feasibility and infeasibility for APSS remain unclear. More specifically, prior work leaves a vexing gap between existing constructions (which rely on some form of synchrony assumption) and existing impossibility results (which rule out fully asynchronous PSS *only* when the reconstruction threshold t coincides with the maximal resilience f). In this work, we find that non-trivial constructions and impossibility results both exist within this gap, giving a more complete picture of (im)possibility for APSS.

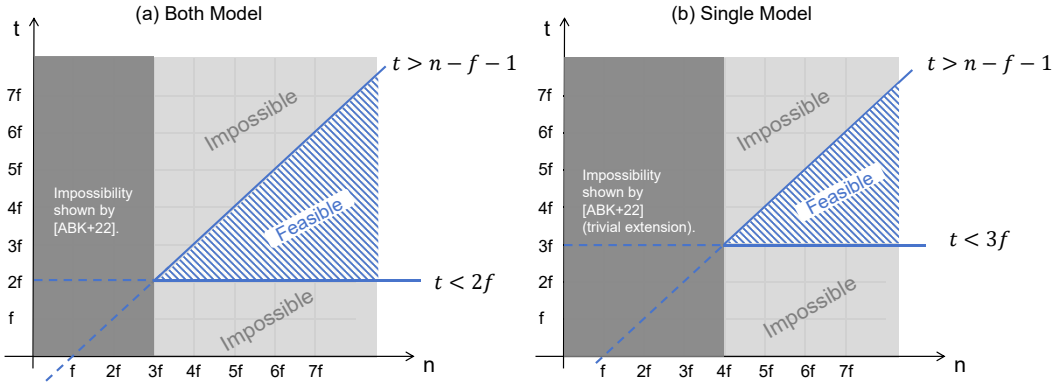


Fig. 1. (Im)possible regimes regarding a partially malicious adversary (horizontal axis: the number of parties n; vertical axis: the reconstruction threshold t).

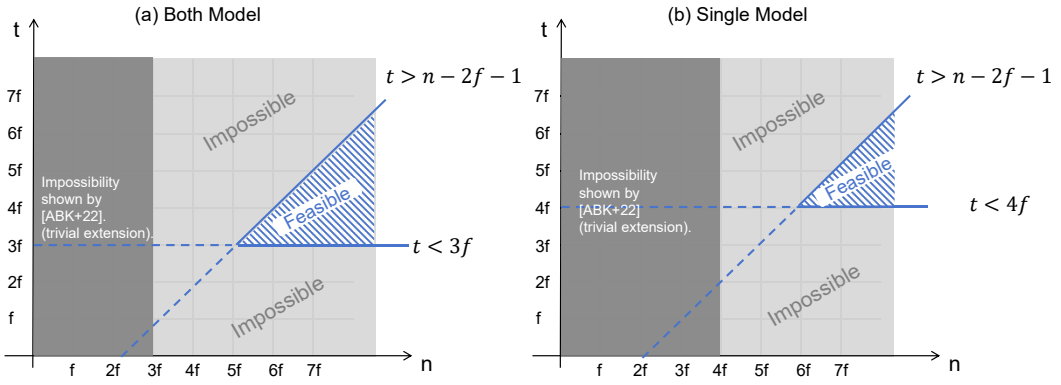


Fig. 2. (Im)possible regimes regarding a malicious adversary (horizontal axis: the number of parties n; vertical axis: the reconstruction threshold t).

1.1 Our Contribution

We conduct a systematic study of APSS to address the aforementioned gap. We begin by carefully modeling adversarial capabilities and formalizing APSS security to capture

Table 1. Impossibility regimes of APSS under certain relationships among n, f, and t (where n is the total number of parties, f is the per-epoch corruption budget of the adversary, and $t+1$ is the number of correct shares necessary to reconstruct the secret).

Corruption accounting models	**Epochs counted by corruption at** $\mathsf{Reshare}^{(\tau\to\tau+1)}$	**Impossible regimes**		**Tight resilience upper bound**	
		Partially mal. adversary	**Malicious adversary**	**Partially mal. adversary**	**Malicious adversary**
BOTH	$\tau, \tau+1$ i.e., current & next	$t < 2f$ -or- $t > n-f-1$	$t < 3f$ -or- $t > n-2f-1$	$f = \lfloor (n-1)/3 \rfloor$, feasible if $t = 2f$	$f = \lfloor (n-1)/5 \rfloor$, feasible if $t = 3f$
SINGLE	τ i.e., only current	$t < 3f$ -or- $t > n-f-1$	$t < 4f$ -or- $t > n-2f-1$	$f = \lfloor (n-1)/4 \rfloor$, feasible if $t = 3f$	$f = \lfloor (n-1)/6 \rfloor$, feasible if $t = 4f$

a previously overlooked interleaving issue arising from multiple Reshare instances, where instances are intended to execute sequentially but may overlap in an asynchronous network. Building on this modeling foundation, we clarify subtle distinctions among a hierarchy of natural adversary models, identify the boundaries of impossibility under each, and present complementary constructions that achieve the optimal resilience allowed by our impossibility results. In more detail, our contributions (also summarized in Table 1) are:

- **Formal modelings of APSS**. We systematically investigate variants of adversary models for APSS protocols and identify subtle yet important distinctions between them:
  - In contrast to the previous impossibility study [2], which only considers a very restricted adversary that can only remove outgoing messages and read states of a corrupted party (called the partially malicious adversary in our terminology[7]), we also introduce a *malicious* adversary that can additionally modify or erase a corrupted party's internal state and incoming messages, as well as forge outgoing messages.
  - We also introduce a hierarchy of two natural models for counting the adversary's per-epoch mobile corruption budget. In the first model (termed the SINGLE model), a Reshare-phase corruption is charged only to the budget of the current epoch. In the second model (termed the BOTH model), a corruption during the Reshare phase is charged to the corruption budgets of both the current and succeeding epochs, which is also natural (and common in synchronous PSS literature [33,29,32,39,7]) since such a Reshare-phase corruption leaks the shares of both epochs.

  We further show (perhaps unsurprisingly, but for completeness, in Section 5.3) that the (partially) malicious adversary in the SINGLE model is strictly stronger than that in the BOTH model.
- **Fine-grained impossibilities of APSS**. We generalize the first impossibility in [2] for $n = 3f+1$ (resp. $n = 4f+1$) under BOTH (resp. SINGLE) model along several axes (summarized in 'Impossible regimes,' Table 1):
  - Unlike ABK+22 [2], our impossibility analysis is not limited to APSS schemes where the reconstruction threshold coincides with the maximum number of corruptions f. Instead, we introduce a separate reconstruction threshold $t \geq f$, such that $t+1$ correct shares suffice to reconstruct the original secret, and use this to show finer-grained impossibility results for a broader class of APSS schemes.[8]
  - In contrast to ABK+22 [2], which only considers (in our terminology) a *partially malicious* adversary, and which does not distinguish between the SINGLE and BOTH corruption accounting models, we systematically study feasibility under the four models that arise from considering each possible combination of corruption accounting model (BOTH/SINGLE) and adversary model (partially malicious/malicious).

[7] We use the term "partially malicious" only to denote this specific read-only adversary with outgoing-message omission. This is not meant to coincide with standard semi-honest or semi-malicious notions in the MPC literature; rather, it identifies the adversarial capabilities used in the impossibility argument of Alexandru et al. [2].

[8] Within our broader landscape of impossibility for APSS, the earlier infeasibility result in [2] can be viewed as treating the special case where $t = f$.

– **Showing the tightness of feasible regimes**. We finally establish constructive proofs for the feasible regimes of APSS by explicit APSS protocols that are secure under each considered model.[9] More specifically,
  - Against partially malicious adversaries, our construction achieves per-epoch resilience $f = \lfloor (n-1)/3 \rfloor$ under the BOTH model with $t = 2f$, and $f = \lfloor (n-1)/4 \rfloor$ under the SINGLE model with $t = 3f$.
  - Against malicious adversaries, our construction achieves per-epoch resilience $f = \lfloor (n-1)/5 \rfloor$ under the BOTH model with $t = 3f$, and $f = \lfloor (n-1)/6 \rfloor$ under the SINGLE model with $t = 4f$.

  Combining both impossibility and feasibility results, we show that the resilience upper bounds implied by our impossibility results are *tight*, yielding a comprehensive picture of (in)feasibility regimes in the presence of a partially malicious adversary (Fig. 1) and a malicious adversary (Fig. 2).

1.2 Related Work

Existing "APSS" need extra synchrony assumptions. Existing attempts at asynchronous proactive secret sharing (APSS) all rely on additional synchrony assumptions (either explicit or implicit) and therefore cannot operate in a fully asynchronous environment. For example, Cachin et al. [14] consider a weakened termination guarantee, whereby resharing completes only if the adversary delivers all messages among honest parties before a corresponding clock tick. Schultz et al. [44] essentially assume a partially synchronous network model instead of a fully asynchronous network. Moreover, several recent studies of APSS [48,28] restrict their security goals to a one-shot resharing between two epochs, rather than addressing a long-running APSS for an unfixed number of resharings.

Existing impossibility in fully asynchronous setting. The landmark result of Alexandru et al. [2] showed impossibility for a specific subclass of APSS protocols with resilience $f = \lfloor (n-1)/3 \rfloor$ and a reconstruction threshold $t = f$ in a setting with a relatively weak partially malicious adversary that can only remove outgoing messages and read internal states of corrupted parties. In contrast, under the same setting of $t = f$, asynchronous verifiable secret sharing (AVSS) *without* proactive security can be realized due to classical results [19,5,14,42], revealing a fundamental feasibility gap between the asynchronous secret sharing schemes with and without proactive security.

PSS in the YOSO model. Recently, another line of work [12,30,20,21,22] has studied proactive refresh of secret sharing in the model now known as the YOSO (You-Only-Speak-Once) model [30]. The YOSO setting primarily aims to tolerate adaptive corruptions across dynamically changing shareholding committees, which are typically (anonymously) sampled from a larger universe of parties. YOSO is particularly suitable for scalable distributed settings, such as blockchains, where a large universe of parties is potentially involved but only small committees are selected per epoch for efficiency, without sacrificing resilience against adaptive corruptions. To date, the YOSO model has been mostly studied in the strictly synchronous setting [12,30,20,21], with some very recent work initiating the study of YOSO-style protocols in *asynchrony* [22]. To our knowledge, none of these works explicitly model and study security in the presence of mobile corruptions; it seems intuitive that the synchronous YOSO protocols would achieve some form of mobile security (or could be straightforwardly adapted), but the situation is much less clear in the asynchronous model, where any PSS scheme (YOSO or not) would face the same impossibility results, technical challenges, and modeling subtleties discussed in our work.

APSS in bulletin-board-aided model. Different from APSS discussed in this work, Levrat et al. [37] recently studied the APSS problem in a different bulletin-board-hybrid setting. In general, they assume a bulletin-board PKI is granted (e.g., modeling an existing blockchain) to enable parties to register their public keys in each epoch. This extra assumption greatly simplifies the APSS problem, as the relaxed setting already allows for tolerating

[9] Note that these APSS protocols are proven secure in our setting of authenticated asynchronous networks, which do not inherently provide communication privacy.

more than $n/3$ corruptions. (The asynchronous YOSO protocols of Damgård et al. [22] mentioned above also uses a ledger, but they construct a ledger with the needed properties as one of their contributions.) Notably, it was a folklore impossibility that no maliciously-secure verifiable secret sharing in the asynchronous setting (even without proactive security) exists if there are $n/3$ corruptions, following the impossibility of asynchronous reliable broadcast with $n/3$ malicious corruptions in the peer-to-peer authenticated setting [13].

SINGLE vs BOTH mobile corruption models in literature. In many synchronous PSS schemes, a party corrupted during the Reshare protocol is charged to the corruption budget of both adjacent epochs [29,3,8] (called BOTH model in this paper). It is also intuitively known that adopting this modeling convention may cause some PSS protocols to enjoy "improved" resilience bounds [37], compared to another natural modeling in which the Reshare-phase corruptions count toward the budget of only a single epoch (called SINGLE model by us). However, to our knowledge, there is no prior work making a formal argument about the relationship between these two natural modelings, and particularly, no established hierarchy among them in the standard peer-to-peer asynchronous setting. In Section 5.3, we provide the first formal separation between the SINGLE and BOTH models, going beyond the previously informal intuition.

2 Technique Overview

This section provides an overview of our systematic approach to characterizing nuanced variations in the security models of APSS protocols and analyzing their (im)possibility results under each model.

Modeling APSS. To capture proactive security in a fully asynchronous network, we first define a notion of epochs that does not rely on a global wall clock. In this simplistic asynchronous modeling (similar to [44]), each party determines epoch transitions locally based on its own state. Syntactically, the epoch number τ is a monotonically increasing local state of each party, and a party advances to a new epoch $\tau+1$ once it completes a share refresh by receiving its updated secret share from the τ-th Reshare protocol. As expected, the notion of local epochs captures the interleaving of multiple Reshare instances, as parties may be in different epochs simultaneously due to variations in network delays in an asynchronous environment.

Accounting models for asynchronous mobile corruptions. As an epoch is defined as a monotonic local state tracking the number of share refreshes by each party, we grant the adversary mobile corruption capability by assigning it a budget of f parties to corrupt per such epoch. However, the accounting of corruption budgets can be modeled in two different ways that are both natural:

- In the first model (the BOTH model) that is commonly considered in many synchronous PSS literature [33,29,32,39,7], a corruption during a Reshare protocol counts toward the adversary's budgets for both the preceding and succeeding epochs, since Reshare is a transition phase between them, and corrupting a party during this phase allows the adversary to obtain both the current epoch's share and the next epoch's share.
- In the second model (the SINGLE model) as an abstraction to the security game from [2], the Reshare protocol is considered part of the current epoch (as each party enters a new epoch only if finishing Reshare), so a corruption during this phase counts only toward the current epoch's budget. Intuitively, this corresponds to a stronger mobile adversary, because corrupting a party during the Reshare phase no longer consumes its corruption budget for the new epoch, allowing the adversary to learn a new epoch's share for free.

Multiple tiers of malicious behaviors. The partially malicious adversary considered in the impossibility proof of Alexandru et al. [2] can only remove outgoing messages and read the internal states of corrupted parties during the epochs in which they are corrupted. While this partially malicious model is interesting in its own right, we additionally formalize a fully malicious adversary that can modify or erase internal states of corrupted parties, forge outgoing messages, and remove/modify incoming messages, as naturally permitted by

malicious behavior and commonly considered in the traditional synchronous PSS setting [33]. The malicious model strengthens the adversary by allowing it not only to observe but also to actively tamper with the protocol execution.

These modeling choices induce a strict hierarchy of adversarial power, resulting in distinct feasibility and resilience thresholds for APSS protocols. Looking ahead, our comprehensive characterization of the (in)feasible regimes under each model also yields, as a byproduct, formal separations between these models that strengthen earlier folklore intuitions (cf. Section 5.3 for details).

Generalized APSS impossibility considering reconstruction threshold. As noted earlier, the impossibility result of Alexandru et al. [2] considers only APSS protocols in which the reconstruction threshold t coincides with the number f of per-epoch corruptions. However, the assumption that $t = f$ is not inherent to APSS, and removing this constraint opens up a wider landscape to study the (im)possibility of APSS. At a high level, we provide two categories of generic attacks, one launched by a partially malicious adversary and the other by a fully malicious adversary, to demonstrate the impossibility of certain regimes under specific relationships among n, f, and t, regardless of APSS designs.

Extending generic partially malicious attack from [2] for $t > f$. To illustrate more fine-grained impossibility regimes and to provide additional insight into the feasible space of APSS, we first consider the (im)possibilities of APSS with an arbitrary reconstruction threshold $t \geq f$. A trivial infeasible regime arises when $t > n-f-1$, since in this case the adversary can corrupt a fixed set of f parties and keep them silent, thereby preventing successful reconstruction; otherwise, reconstruction would require fewer than $t+1$ shares, violating the definition of reconstruction threshold t.

We then demonstrate another impossibility relationship $t < 2f$ (resp. $t < 3f$) in the presence of a partially malicious adversary under the BOTH (resp. SINGLE) model, by showing that the adversary can violate either privacy or liveness in this regime. To show this, we extend the partially malicious attack of Alexandru et al. to arbitrary $t \geq f$. At a high level, the attack works as follows.

The attacker selectively delays certain messages, so that only $t+1$ parties advance to the current "frontier" epoch, τ, and other parties remain in earlier epochs before $\tau - 1$. The adversary can corrupt f parties out of the $t+1$ frontier parties. In the BOTH model, the adversary learns the shares of these f frontier parties. In the SINGLE model, the adversary learns $2f$ frontier shares in total (where the f additional shares come from its corruptions during the Reshare protocol in epoch $\tau - 1$, as allowed by the SINGLE model). Even worse, the adversary can remove the outgoing messages of the f parties corrupted during epoch τ, causing the frontier Reshare protocol to lack sufficient participants to complete. This leads to the following dilemma:

(i) If less than f delayed parties can compute their refreshed secret shares so as to advance to the frontier epoch τ, liveness is already violated. This is because, out of $t+1$ parties staying in the frontier epoch, f can be corrupted and behave as if they have crashed. Hence, only at most t honest parties remain to execute the frontier epoch τ's Reshare protocol. This is insufficient in number for any honest party to advance to epoch $\tau+1$ as $t+1$ are needed for resharing/reconstruction correctness.

(ii) As implied by the above liveness-violation scenario, the frontier Reshare protocol in epoch τ requires at least f delayed parties to be able to compute their refreshed shares for the frontier epoch τ to maintain liveness. Recall that the $t+1$ frontier parties have erased their secret shares and all other private states related to epoch $\tau - 1$, as required for meaningful proactive security. The remaining delayed parties have not yet entered the epoch $\tau - 1$. Consequently, the delayed parties must rely solely on their current local states together with already-sent (but still in-transit) messages to compute their shares for epoch τ.

Leveraging this, the adversary can strategically corrupt the delayed parties at epochs prior to τ. In this manner, it can observe their internal states for those epochs together with all in-transit messages sent to those parties. From the above discussion, if liveness is to be maintained, these states and messages allow the adversary to obtain at least f additional secret shares for the frontier epoch τ.

In summary, a partially malicious adversary can in total learn $2f$ (resp. $3f$) shares for epoch τ under the BOTH (resp. SINGLE) model, where f shares are obtained from delayed parties and f (resp. $2f$) directly from frontier parties. Thus $2f \leq t$ (resp. $3f \leq t$) if both liveness and privacy are held, implying another impossible regime $t < 2f$ (resp. $t < 3f$).

Putting the impossible regimes $t > n - f - 1$ and $t < 2f$ (resp. $t < 3f$) together under the BOTH (resp. SINGLE) model, we can derive the resilience upper bound $f = \lfloor \frac{n-1}{3} \rfloor$ (resp. $f = \lfloor \frac{n-1}{4} \rfloor$) of APSS against partially malicious adversaries under the BOTH (resp. SINGLE) model. Here, the difference in resilience between the SINGLE and BOTH models can be understood as in the SINGLE model, the adversary can additionally learn f shares of the succeeding epoch "for free" by consuming only the corruption budget of the current epoch, necessitating larger t and n (relative to f) to counter this additional leakage.

New generic attack to show impossibility under malicious adversary. We next consider APSS facing a stronger malicious adversary. The attack hinges on the malicious attacker's additional capabilities, namely, that for any party corrupted during epoch τ, the adversary can send forged outgoing messages, delete/modify incoming messages, and erase/modify the current secret shares.

Leveraging these additional malicious capabilities, we establish a stronger impossibility regime $t > n - 2f - 1$ in the malicious setting, replacing the trivial impossible condition of $t > n - f - 1$. The generic attack works by totally removing f parties' secret shares (and all messages needed by these parties to advance to a new epoch). Specifically, the adversary lets all n parties initially stay in the same epoch, out of which f are corrupted. As the adversary is malicious, it can totally isolate these f corrupted parties from interacting with other $n - f$ parties in the Reshare protocol by removing their incoming messages and also removing their secret shares. Critically, the adversary can still allow the remaining $n - f$ parties to advance to a new epoch by, e.g., delivering all Reshare-protocol messages. Now, $n - f$ parties enter the new epoch, thus leaving the f corrupted parties completely "trapped" in the last epoch. One might also suggest that the $n - f$ parties advancing in the new epoch can help these f to advance, but recall that these $n - f$ parties have removed their previous shares and all private states used during the previous Reshare protocol (per requirement of meaningful proactive security); moreover, the adversary also obtains a fresh budget to corrupt and mute f of these $n - f$ parties (after releasing f corrupted parties trapped in the last epoch). So, to either help the f parties trapped in the last epoch to advance or have the new Reshare protocol to be completed, we require the remaining $n - 2f$ honest parties in the new epoch alone to be sufficient in number (i.e., $n - 2f$ at least $t + 1$ as the minimal number for resharing).

Given the strengthened impossibility regime $t > n - 2f - 1$ in the malicious setting, we further leverage its attack strategy, namely, the complete removal of an additional f parties' frontier shares, to extend the previously described partially malicious attack that establishes the impossibility of $t < 2f$ (resp. $t < 3f$) under the BOTH (resp. SINGLE) model. This extension yields a stronger malicious-setting impossibility of $t < 3f$ (resp. $t < 4f$) under the BOTH (resp. SINGLE) model, as more delayed parties ($2f$ instead of f) are required to obtain frontier shares (to restore liveness), and unfortunately, causing more frontier shares to be leaked to the asynchronous mobile adversary. Combining these bounds yields the resilience upper bound $f = \lfloor \frac{n-1}{5} \rfloor$ (resp. $f = \lfloor \frac{n-1}{6} \rfloor$) of APSS against malicious adversaries under the BOTH (resp. SINGLE) model.

Constructive proofs (by explicit APSS protocols) to show tightness of impossibilities. Finally, we show that the above impossibility regimes are tight by providing APSS constructions that securely achieve optimal resilience in each adversary model.

Partially malicious APSS scheme. Our partially malicious APSS construction lets the first $t+f+1$ out of n parties obtain initial t-degree Shamir secret shares and then participate in Reshare protocols. (We will soon explain why $t + f + 1$ parties instead of n.) Our partially malicious Reshare construction (run by the first $t + f + 1$ parties) begins with an ephemeral encryption-key publish phase, during which the parties agree on a set of fresh ephemeral encryption keys from at least $t + 1$ parties. This is followed by a resharing phase, in which each party computes subshares of its current share and constructs a resharing transcript

(where each subshare is either encrypted under the corresponding recipient's ephemeral key or simply in plaintext if the recipient's encryption key was not agreed upon). Finally, the parties agree on resharing transcripts from $t+1$ distinct parties. Once these transcripts are agreed upon, every party can derive its new share from the collected subshares. In greater detail, Reshare proceeds as follows:

(i) (*Ephemeral-PK Publish and Agreement*). For each $\mathcal{P}_i$ (where $i \in [1, t+f+1]$) in epoch τ, it samples a fresh ephemeral public/secret key pair $(\mathsf{esk}_i, \mathsf{epk}_i)$ for public-key encryption (PKE), multicasts the public encryption key epk_i, and then participates in an asynchronous agreement. [10] This agreement protocol requires a quorum of $t+1$ honest parties in order to terminate, and we will soon discuss why sufficiently many honest parties eventually enter the epoch τ to ensure this. The agreement protocol then confirms a set T_{epk} of at least $t+1$ ephemeral encryption keys $\{\mathsf{epk}_j\}$ from distinct parties.

(ii) (*Resharing-Transcript Publish and Agreement*). After receiving the agreed-upon set T_{epk} of ephemeral encryption keys, $\mathcal{P}_i$ (where $i \in [1, t+f+1]$) samples a random polynomial $\phi_i(X)$ with degree up to t and $\phi_i(0) = s_i^{(\tau)}$, where $s_i^{(\tau)}$ is $\mathcal{P}_i$'s secret share in epoch τ, and $\phi_i(j)$ is a subshare of $s_i^{(\tau)}$ corresponding to $\mathcal{P}_j$. Then, $\mathcal{P}_i$ prepares and multicasts a resharing transcript $c_i \leftarrow \{c_{i,j}\}_{j\in[1,t+f+1]}$, where $c_{i,j} = \mathsf{Enc}(\mathsf{epk}_j, \phi_i(j))$ iff $\mathsf{epk}_j \in T_{\mathsf{epk}}$. Otherwise $c_{i,j} = \phi_i(j)$ is added to the resharing transcript c_i in the clear, i.e., without encrypting it.[11]
Then another agreement protocol is executed to confirm a set $T_s = \{c_j\}$ consisting of $t+1$ resharing transcripts from distinct $\mathcal{P}_j$. (The agreement again requires a quorum of $t+1$, and we will soon explain why there are sufficient honest parties to make it terminate).

(iii) (*New-Share Interpolation and Erasure*). Finally, upon receiving the agreed-upon set T_s, $\mathcal{P}_i$ parses each $c_j \in T_s$ to get the i-th item $c_{j,i}$, which directly carries $\phi_j(i)$ in the plaintext or encrypts $\phi_j(i)$ and can be decrypted. Now, $\mathcal{P}_i$ can get $t+1$ such $\phi_j(i)$ from distinct $\mathcal{P}_j$, and it can interpolate these points to get a t-degree polynomial $\phi'_i(X)$, where $\phi'_i(j) = \phi_j(i)$ for each j. Finally, $\mathcal{P}_i$ sets its new secret share $s_i^{(\tau+1)} \leftarrow \phi'_i(0)$ and erases all outdated private states including $s_i^{(\tau)}$, esk_i, other intermediate states such as $\phi'_i(X)$ and $\phi_i(X)$, and any randomness used so far in the execution.

Instantiating asynchronous agreement despite mobile adversary. First, we remark that asynchronous agreement in our protocol can be instantiated in the full-information setting [35], relying solely on authenticated point-to-point channels without requiring participants to maintain any long-term secrets. More precisely, the full-information setting allows the adversary to observe all parties' agreement-related states and messages. This choice simplifies the presentation of our protocols and yields more concise security proofs, allowing us to focus on establishing tight feasibility results for APSS. Otherwise, if the underlying agreement protocols rely on threshold cryptosystems, one would need to proactively reshare long-term secrets to maintain security across epochs. By using the full-information asynchronous agreement, we avoid this complication and simplify our tightness argument.

Liveness argument. We now circle back to argue liveness, given full-information asynchronous agreement. In the above Reshare protocol, there are two asynchronous agreement instances, each requiring a quorum of $t+1$ to terminate. Their termination can be argued inductively. In epoch 0, clearly $t+f+1$ parties will participate in these agreements. Since at most f parties can be corrupted in epoch 0, at least $(t+f+1)-f = t+1$ of them remain honest in the epoch. Hence, both agreements can terminate in epoch 0. Another important aspect of our construction is: even if a party fails to publish its ephemeral encryption key in epoch 0, it still receives its corresponding subshares in plaintext, thus advancing to epoch

[10] Note that we use an asynchronous agreement that relies only on authenticated channels and does not assume a PKI or any threshold cryptosystem setup. As a result, it avoids long-term secret material altogether, eliminating the risk of long-term key exposure to a mobile adversary. So as long as sufficient currently-honest parties are participating, such an agreement can finish (even if all parties were once corrupted).

[11] To be precise, we encrypt $\phi_i(j)$ when $\mathsf{epk}_j \notin T_{\mathsf{epk}}$ using a default encryption key with a publicly-known default decryption key, just for presentation brevity.

1 with its refreshed share. Consequently, all $t + f + 1$ participating parties eventually enter epoch 1. Inductively, the same argument implies that these $t+f+1$ parties eventually enter epoch 2, epoch 3, and so on.

Privacy argument. Next, we comment on the issue of privacy resilience, and specifically why this construction yields tight privacy resilience guarantees. Careful readers may notice two seemingly counterintuitive design choices: (i) only the first $t + f + 1$ parties get the initial shares and participate in Reshare; and (ii) parties whose ephemeral encryption keys are not agreed upon will receive their refreshed shares in the clear to facilitate liveness. The first design choice is motivated by the generic partially malicious attack used in proving impossibility, where the adversary lets only $t+1$ parties advance to the frontier epoch while delaying the remaining $n - t - 1$ parties. For tight resilience, the adversary should learn at most f frontier shares from these $n - t - 1$ delayed parties. Accordingly, we exclude $(n - t - 1) - f$ parties from holding initial shares and from participating in resharing. This guarantees that among the $n - t - 1$ delayed parties, at most f can derive frontier shares, in contrast to learning $n - t - 1$ shares from $n - t - 1$ delayed parties. The second choice does not weaken tight resilience either: in the generic attack underlying impossibility, the adversary can anyhow learn f shares from delayed parties. Hence, revealing their refreshed shares in the clear (to facilitate liveness) does not provide any additional advantage to the adversary.

Tightness of resilience. Combining both privacy and liveness arguments, our simple partially-malicious-secure APSS achieves a tight resilience against a partially malicious mobile adversary. In particular, consider the following example to see the tightness. Let $n = 4f + 1$ and a reconstruction threshold $t = 3f$, and let all $4f + 1$ parties receive initial shares and run resharing protocols. For any epoch τ, there are eventually $3f + 1 = t + 1$ honest parties entering there, making resharing terminate and advancing to epoch $\tau + 1$; moreover, a partially malicious adversary under SINGLE mobile corruption model can learn at most $3f$ shares in the epoch τ: (i) f shares directly from parties corrupted in epoch τ; (ii) another f shares from parties corrupted in the previous epoch $\tau - 1$, as allowed by SINGLE model; (iii) the final f shares from parties that were delayed in even earlier epochs, whose undelivered messages directly carry their epoch-τ shares. That said, a tight resilience $f = \lfloor (n - 1)/4 \rfloor$ is realized in the partially malicious SINGLE model.

Malicious APSS scheme. For a malicious adversary, we must additionally handle (i) arbitrarily malformed messages associated with epoch τ that are sent by parties corrupted at epoch τ, and (ii) the complete erasure/modification of a party's secret share and incoming messages associated with epoch τ (during which the party is corrupted). We address these challenges as follows.

First, we let each party maintain an auxiliary Pedersen polynomial commitment to each epoch's secret-sharing polynomial, which is also updated consistently at every Reshare while the secret shares are refreshed. To enable such updates during epoch transitions, each resharing of the current share is not only encrypted under the recipient's corresponding ephemeral public encryption key, but is also accompanied by (i) a Pedersen polynomial commitment to the resharing polynomial and (ii) appropriate zero-knowledge proofs attesting to the consistency between each reshare's ciphertext and the Pedersen committed polynomial. Together, these components constitute a publicly verifiable secret sharing (PVSS) that certifies the correctness of the resharing with respect to the currently committed share, from which the new auxiliary polynomial commitment for the refreshed secret shares can also be derived.

Moreover, it is critical to note that in the malicious setting, our construction chooses the first $t+2f+1$ out of all n parties to get the initial shares and participate in the subsequent resharing protocols. This is necessary to preserve liveness, as indicated by our generic malicious attack: the adversary may completely erase the internal share and incoming messages of up to f parties, thereby trapping them in a preceding epoch, while simultaneously corrupting up to f additional parties during the current Reshare epoch. As a consequence, any secure construction must remain live even when $2f$ parties are unavailable to the Reshare protocol. To satisfy this constraint, we need $t + 2f + 1$ participants.

Finally, for a party that was corrupted during some epoch τ and remains trapped in that epoch even after the adversary releases it (due to omitted messages associated with

τ), we provide an alternative recovery mechanism for epoch advancement: such a party can advance by receiving a sufficient number of resharing messages from a future epoch, which the adversary can no longer omit. As a result, once the adversary delivers enough messages at the frontier epoch to allow parties there to advance, any party trapped in an earlier epoch can eventually catch up to the frontier epoch. This ensures liveness, provided that the frontier Reshare protocol has $t+1 = n-2f$ participating parties, out of which at least $n-3f$ parties can eventually advance into an even newer epoch as honest ones, and other f parties that may be trapped in the previous epoch due to message omission and state erasure can also catch up, making the new frontier epoch to have $n-2f$ honest participants again.

Putting the above enhancements together, a maliciously secure APSS with tight resilience is realized, i.e., achieving tight resilience $f = \lfloor (n-1)/5 \rfloor$ (resp. $f = \lfloor (n-1)/6 \rfloor$) in the BOTH (resp. SINGLE) model against a malicious mobile adversary, when t is parameterized as $3f$ (resp. $4f$).

3 Modeling

3.1 System and Adversary in Overview

Participating parties. We assume a distributed system that consists of a designated dealer $\mathcal{D}$ (that remains honest) and a set of n parties $\{\mathcal{P}_i\}_{i\in[n]}$. Each party has a unique identifier (e.g., $\mathcal{P}_i$ for some $i \in [n]$) and is modeled as a probabilistic polynomial-time Interactive Turing Machine (ITM), with an outgoing tape and an incoming tape representing the network I/O interfaces. For brevity, we will say that a party sends a message whenever the ITM writes a message to its outgoing tape, and that a party receives a message (or a message is delivered) whenever a message is written to the incoming tape for that ITM.

We also assume honest parties are capable of secure erasure, i.e., that they can erase values from memory so that an adversary that corrupts the party at a later point learns nothing about the erased value. Such a secure erasure assumption is standard and often considered necessary in proactive secret sharing [33,32].

f-limited mobile adversary. We consider a probabilistic polynomial-time (PPT) adversary $\mathcal{A}$ that can simultaneously corrupt up to f out of the n parties. Moreover, we consider that the adversary can perform so-called mobile corruption [40,33], meaning that it can release a previously corrupted party and subsequently corrupt another party of its choice. (We will formalize our modeling of such mobile corruptions in later subsections.) Informally speaking, we will distinguish between two types of adversaries, characterized as follows:

- *Partially Malicious adversary.* A partially malicious adversary $\mathcal{A}^{\mathsf{part}}$ obtains read access to the full internal state of each corrupted party, including its secret share, private keys, and any other internal variables belonging to the corrupted party. (Any internal variables that were securely erased prior to the party being corrupted are not visible to the adversary.) $\mathcal{A}^{\mathsf{part}}$ retains access to any values it reads, even if the party later becomes uncorrupted (say, by copying the data to another machine). Moreover, a partially malicious adversary can also drop messages sent from a corrupted party.
- *Malicious adversary.* A malicious adversary $\mathcal{A}^{\mathsf{byz}}$ has all the same abilities as a partially malicious adversary, and further can send arbitrary messages on behalf of a corrupted party, as well as delete/modify the internal states and incoming messages of a party during its corruption.

A malicious adversary is certainly at least as strong as its partially malicious counterpart, and intuitively, we might predict that a malicious adversary is strictly stronger. Looking ahead, we will revisit their precise malicious abilities in the context of mobile corruptions later in Section 3.4, discuss (im)possibilities under each adversary model in Section 4, and reveal that a malicious adversary is indeed strictly stronger in Section 5.1 (where a partially-malicious-secure protocol is provided under the impossible regime of a malicious adversary).

Asynchronous network (*authenticated but not private*). We assume that the parties are connected by pairwise authenticated and reliable asynchronous point-to-point communication channels [19,9,11,16,15,1]. No upper bound is assumed on message delivery latency.

To model this, we let all communications go through the adversary (i.e., it has access to all parties' outgoing tapes as well as the writing permission of each party's incoming tape, and has full control over the scheduling and delivery of messages), therefore enabling the adversary to arbitrarily delay and reorder messages after seeing them in the clear.

A standard authenticated asynchronous network is reliable in the sense that messages sent between two uncorrupted parties must be eventually delivered without tampering. However, in the mobile corruption setting, each party may be corrupted during some periods and uncorrupted during others. Jumping ahead, we extend the reliability of communication as follows in the mobile corruption model: each message is associated with a given epoch τ (which will be formally defined later) and is guaranteed to be eventually delivered without tampering if both the sending and receiving parties are uncorrupted during the same epoch τ. This adapted reliable channel aligns with our adversary model: when a party is corrupted during an epoch τ, a partially malicious adversary may drop all outgoing messages from that party, whereas a malicious adversary may additionally drop all incoming messages sent to that party and also belonging to the epoch τ.

3.2 Syntax

Asynchronous proactive secret sharing. We begin by defining syntax for an asynchronous proactive secret sharing (APSS) scheme. We defer the definition of security for APSS (Definition 4) momentarily, until after we have introduced the mobile corruption models (Definitions 2 and 3).

We adapt the standard PSS syntax to APSS, which begins with a dealer running a sharing algorithm and distributing the shares, then proceeds through a sequence of resharing protocols to proactively refresh secret shares, and finally concludes with a reconstruction protocol to recover the original secret.

Definition 1 (Syntax of APSS). *An (n, f, t)-asynchronous proactive secret sharing (APSS) scheme consists of algorithms* Share*,* $\mathsf{Reshare}$*, and* $\mathsf{Reconstruct}$*. Throughout, we assume each party $\mathcal{P}_i$ has internal state that includes its current* epoch number τ *and a boolean flag* $\mathsf{handoff}$*; we say that P_i is* staying in epoch τ *exactly while its epoch number is set to τ and has not yet been set to any higher value. The algorithms have the following syntax and behavior:*

- Share *is an algorithm run by a dealer $\mathcal{D}$ that takes as input a secret $s \in \mathbb{F}$ and outputs initial shares $(s_1^{(0)}, \ldots, s_n^{(0)})$, along with some public auxiliary information $\mathsf{aux}^{(0)}$ and public parameters pp. Each $\mathcal{P}_i$ is exclusively given a secret share $s_i^{(0)}$. Upon receiving its initial share, a party initializes its epoch number to 0 and sets* $\mathsf{handoff}$ *to* false*.*
- $\mathsf{Reshare}$ *is an interactive protocol. Each instance of* $\mathsf{Reshare}$ *is associated with an identifier τ and executed by any set of at least $t+1$ parties providing their valid inputs at epoch number τ. Upon a party $\mathcal{P}_i$ at epoch τ provides its secret share $s_i^{(\tau)}$ together with auxiliary information $\mathsf{aux}^{(\tau)}$ as input to the* $\mathsf{Reshare}$ *protocol associated with identifier τ,*[12] *it sets* $\mathsf{handoff}$ *to* true*. We say such a party $\mathcal{P}_i$ with* $\mathsf{handoff} = \mathsf{true}$ *in the* $\mathsf{Reshare}$ *phase, and denote the* $\mathsf{Reshare}$ *associated with epoch τ by* $\mathsf{Reshare}^{(\tau \to \tau+1)}$ *or* $\mathsf{Reshare}^{(\tau)}$*. A party $\mathcal{P}_i$ at epoch $\tau' \leq \tau$ can receive the refreshed secret share $s_i^{(\tau+1)}$ and the updated public auxiliary information $\mathsf{aux}^{(\tau+1)}$ as output from $\mathsf{Reshare}^{(\tau \to \tau+1)}$;*[13] *upon that, $\mathcal{P}_i$ sets its epoch number to $\tau+1$ and sets* $\mathsf{handoff} = \mathsf{false}$*, and securely erases all other internal states (including intermediate states of all finished* $\mathsf{Reshare}$ *protocols), with only exception of $s_i^{(\tau+1)}$,* $\mathsf{aux}^{(\tau+1)}$*,* pp*, the epoch number, and the* $\mathsf{handoff}$ *flag* [14]*.*

[12] Once a party $\mathcal{P}_i$ enters epoch τ, it can be activated to provide input to $\mathsf{Reshare}^{(\tau \to \tau+1)}$, either after a fixed local time, or by sending itself a special message and waiting for the message to be delivered (equivalent to waiting for an eventual activation command from the adversary). This distinction is inconsequential in our setting, and we will choose the former way in our security games for brevity.

[13] We model that a party at any epoch $\tau' \leq \tau$ may receive $s_i^{(\tau+1)}$ and $\mathsf{aux}^{(\tau+1)}$ from $\mathsf{Reshare}^{(\tau \to \tau+1)}$ as output and therefore advance to epoch $\tau+1$, even if it does not provide an input to $\mathsf{Reshare}^{(\tau \to \tau+1)}$, to capture certain APSS schemes that can be designed to allow a party "skip" an epoch in response to the asynchronous adversary that arbitrarily delays/reorders messages.

[14] Secure erasure of out-dated secret states is basic for meaningful proactive security.

– Reconstruct *is an interactive protocol. Each instance of* Reconstruct *is associated with an epoch* τ *and run by a minimum number of* $t+1$ *parties each with current epoch number* τ*, such that each party* $\mathcal{P}_i$ *inputs its secret share* $s_i^{(\tau)}$ *(possibly together with auxiliary information* $\mathsf{aux}^{(\tau)}$*), such that some designated parties (e.g.,* $\mathcal{P}_1, \cdots, \mathcal{P}_n$*, without loss of generality, throughout the paper) receive an output* $s' \in \mathbb{F}$ *or an error symbol* $\perp$*.*

Remarks on Syntax. For brevity of notations, in the above syntax, each secret share, for example $s_i^{(\tau)}$, can be regarded as a set of exclusive secret states held by the party $\mathcal{P}_i$, which not only contains a (Shamir) secret share but also includes other necessary secrets (e.g., a private encryption key or the hiding secret of cryptographic commitments) if applicable. Moreover, note that we continue the convention from earlier sections of distinguishing the reconstruction threshold and the corruption threshold: throughout, t represents a reconstruction threshold such that $t+1$ parties are needed to reconstruct/reshare the secret, while f denotes the corruption limit.

Similar to [44], the syntax makes explicit a natural notion of a party's local *epoch*. The goal of the APSS scheme, informally, is to ensure that the secret remains confidential even if the adversary can rotate its set of corrupted parties over time, as long as the adversary never corrupts more than a threshold number of parties within any single period (known as an *epoch* in literature). However, the notion of "periodically", which is trivial to define in the synchronous setting due to a global wall clock, becomes ambiguous in the asynchronous setting. To formalize epochs in the asynchronous setting, we augment the syntax with a local monotonic counter that records how many times secret refreshment has been completed. An important consequence of such a locally defined epoch is that an asynchronous adversary can "trap" a party $\mathcal{P}_i$ in an old epoch by delaying messages sent to $\mathcal{P}_i$, while simultaneously allowing other parties to advance to a more recent "frontier" epoch by delivering messages to them.

Notably, the modeling effect of local epochs is intended to capture a previously overlooked issue—multiple conceptually sequential protocols might still interleave each other in a fully asynchronous network.

Our syntax also explicitly defines another local state handoff, to reflect whether a party is staying in the middle of the Reshare phase or not. This notion will be critical to our fine-grained mobile corruption model. We will discuss the corruption models in the next subsection, but to give a brief intuition: a mobile adversary can move its corruptions over parties staying at different local epochs, subject to a per-epoch corruption budget. If a party is not participating in resharing, then no question corrupting that party should count towards the budget for that party's current epoch. However, if a party is in the process of a handoff (i.e., in the middle of resharing), the adversary can learn intermediate values related to that party's shares in *both* adjacent epochs, and, by syntax definition, obtains the shares of both epochs. We therefore explicitly define a handoff flag in our APSS syntax to distinguish if a party stays in the Reshare protocol or not, enabling us to capture this subtlety in our mobile corruption models by charging the resharing-phase corruptions to the per-epoch budgets in different ways.

3.3 Accounting for Mobile Corruptions

For an APSS scheme with the syntax defined above, we consider a mobile adversary in the long-term setting [40,33]. As participants transition from one epoch to the next, the adversary may release a party corrupted in the previous epoch and, in turn, corrupt a party that was honest, as long as the adversary never controls more than f parties globally or within any single epoch. This model nicely captures two real-world phenomena: first, an attacker might gradually gain control over new participants; second, over time, some previously corrupted parties may recover from adversarial influence (e.g., through anti-virus scans, software patching, or security upgrades) and thus be considered honest again.

Epoch-wise corruption budget. It is clear that, without any restriction on the adversary's speed in rotating the set of corrupted parties, achieving meaningful proactive secret sharing becomes impossible, as the adversary can rapidly compromise participants one by one to

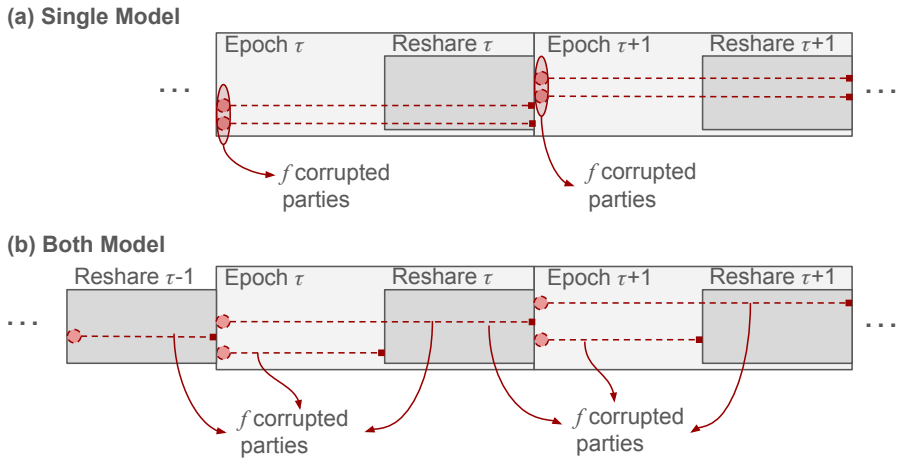


Fig. 3. Examples for SINGLE and BOTH models of mobile corruptions in APSS.

obtain all secret shares almost simultaneously. However, mobile corruption is not intended to model such an overly pessimistic scenario, and existing practice (in the synchronous setting) restricts the number of corrupted parties on a per-epoch basis.

Even at the epoch level, there are subtleties to defining the adversary's capabilities. One possible approach is to consider that the adversary is assigned a separate corruption budget $f^{(\tau)} = f$ associated with each epoch τ, such that it can at most corrupt up to f parties while these parties are staying in epoch τ, as illustrated in Figure 3 (a) and defined in the following.

Definition 2 (SINGLE accounting model for mobile corruption). *Before the first party $\mathcal{P}_i$ enters the epoch τ, the adversary $\mathcal{A}$ must have specified a fixed set of parties $\mathbb{C}^{(\tau)}$, such that $|\mathbb{C}^{(\tau)}| \leq f$. During the subsequent execution, $\mathcal{A}$ can only corrupt parties at epoch τ subject to the following constraints:*

- *if party $\mathcal{P}_i$ is corrupted while staying in epoch τ, then $\mathcal{P}_i \in \mathbb{C}^{(\tau)}$;*
- *$\mathcal{A}$ corrupts at most f parties at any time.*

Note that under the above mobile corruption model, a party $\mathcal{P}_i$ corrupted at any time during epoch τ is counted solely toward the corruption budget of a *single* epoch τ. This applies even if $\mathcal{P}_i$ is corrupted while executing the reshare protocol in epoch τ, in which case the adversary could simultaneously obtain both $s^{(\tau)}$ and $s^{(\tau+1)}$ due to syntax definition. The term "SINGLE", therefore, refers to the fact that such corruption during the resharing protocol is only applied to the corruption budget of a single epoch.

Although the SINGLE model appears to be a natural treatment, many existing studies adopt an alternative approach in which a corruption during the resharing phase is counted against the budgets of both adjacent epochs [33,2], defined as follows and illustrated as in Figure 3 (b).

Definition 3 (BOTH accounting model for mobile corruption). *Before the first party $\mathcal{P}_i$ enter each epoch τ, the adversary $\mathcal{A}$ chooses a set of parties $\mathbb{C}^{(\tau)}$ and another set of parties $\mathbb{C}^{(\tau)}_{\mathsf{reshare}}$, such that $\mathbb{C}^{(\tau)}_{\mathsf{reshare}} \subseteq \mathbb{C}^{(\tau)}$ and $|\mathbb{C}^{(\tau-1)}_{\mathsf{reshare}} \cup \mathbb{C}^{(\tau)}| \leq f$. During the subsequent execution, $\mathcal{A}$ can only corrupt parties at epoch τ subject to the following constraints:*

- *if party $\mathcal{P}_i$ is corrupted while staying in epoch τ, then $\mathcal{P}_i \in \mathbb{C}^{(\tau)}$;*
- *if party $\mathcal{P}_i \in \mathbb{C}^{(\tau)} \setminus \mathbb{C}^{(\tau)}_{\mathsf{reshare}}$, the adversary shall release $\mathcal{P}_i$ from corruption before its flag* handoff *becomes true while $\mathcal{P}_i$ stays in epoch τ. Namely, the adversary can corrupt a party $\mathcal{P}_i$ while it executes the* Reshare *protocol of epoch τ, only if it is specified in the set of $\mathbb{C}^{(\tau)}_{\mathsf{reshare}}$.*
- *$\mathcal{A}$ corrupts at most f parties at any time.*

The formal SINGLE and BOTH accounting models for mobile corruptions will be presented in the next subsection, as part of the security definitions for APSS. Moreover, a mobile adversary due to the SINGLE model is intuitively stronger than that of the BOTH model. In particular, under the SINGLE model, an adversary corrupting a party during the Reshare protocol of the current epoch can effectively obtain the secret share for the succeeding epoch "for free", whereas under the BOTH model, such leakage must be charged against the adversary's corruption budget in both the current and succeeding epochs. For completeness, we will later prove that this intuitive separation indeed holds in Section 5.3.

3.4 Security Modeling

We now define the desired security properties (privacy, liveness, correctness) for APSS. Each is defined in terms of a game between an adversary and a challenger.

Definition 4 (Security of APSS). *An (n, f, t)-asynchronous proactive secret sharing scheme run among n distributed parties is secure against a PPT f-limited adversary $\mathcal{A}$ under the SINGLE (resp. BOTH) mobile corruption model, if it satisfies the following properties, even after T epochs (where T could be unpredictable as chosen by the adversary $\mathcal{A}$):*

- *Privacy: $\mathcal{A}$'s advantage in Privacy game G_S^P (resp. G_B^P) is negligible.*
- *Correctness: $\mathcal{A}$'s advantage in Correctness game G_S^C (resp. G_B^C) is negligible.*
- *Liveness: $\mathcal{A}$'s advantage in Liveness game G_S^L (resp. G_B^L) is negligible.*

Informally, if an adversary's advantage in the correctness game is negligible, then even after T epochs of sequential resharings, the secret reconstructed in the T-th epoch remains identical to the original secret. Similarly, if an adversary's advantage in the liveness game is negligible, then even after T epochs of resharing, all parties still can receive some value from the reconstruction protocol associated with the T-th epoch, conditioned on the adversary eventually delivering all messages from honest senders associated with an epoch number $\tau \leq T$. The correctness and liveness properties complement each other to ensure that all parties can eventually reconstruct the originally shared secret. Moreover, if a mobile adversary has only a negligible advantage in the privacy game, then it is infeasible for it to learn any information about the secret shared by an honest dealer, before entering the reconstruction phase.

Note that in all the above security games, we model the dealer to be honest, mainly to capture the security properties of the Reshare and Reconstruct protocols. Moreover, in many practical applications, such as key escrow, any security loss caused by a malicious dealer would ultimately hurt the dealer itself. Our security games, therefore, focus on the periodic resharing phase after the initial key shares have been correctly distributed.

REMARKS ON MODELING CHOICES. Our security game definitions precisely capture the adversarial capabilities described in the earlier subsections. In addition, they incorporate several subtle yet important aspects:

One critical aspect of our modeling is that even a malicious adversary cannot directly manipulate the epoch of a corrupted party, making the epoch number act as a monotonically increasing local "time". Nevertheless, both partially malicious and malicious adversaries can indirectly influence epochs of corrupted and honest parties by selectively delivering or delaying messages that trigger epoch advancement. For example, an adversary may "trap" a party $\mathcal{P}_i$ in an old epoch by delaying messages to it, while allowing another set of parties to advance to a more recent "frontier" epoch by delivering messages to them. We emphasize that this "interleaving" effect (that many parties may simultaneously stay in different epochs) is central to our modeling objectives in an asynchronous network.

Moreover, inspired by the synchronous setting in which a global-clock-driven epoch advancement can automatically clean corruptions [40] upon a clock tick, we similarly model a corrupt party as being cleaned and recovered upon its entry into a new epoch. However, slightly different from the synchronous case, the asynchronous adversary is also allowed to release a corruption earlier, before the corrupted party enters a new epoch, thereby freeing a global budget to corrupt another party that possibly stays in a different epoch.

Privacy Game G_S^P (for APSS under SINGLE model):

The game proceeds as follows between a PPT adversary $\mathcal{A}$ and a challenger $\mathcal{C}$.

1. $\mathcal{A}$ chooses two distinct secrets $s_0, s_1 \in \mathbb{F}$ and gives them to $\mathcal{C}$.
2. The challenger $\mathcal{C}$ chooses a secret random bit $b \leftarrow_\$ \{0,1\}$, then runs $(s_1^{(0)}, \ldots, s_n^{(0)}; \mathsf{aux}^{(0)}, \mathsf{pp}) \leftarrow \mathsf{Share}(s_b)$, and sends $\mathsf{aux}^{(0)}$ and pp to the adversary $\mathcal{A}$.
3. The challenger $\mathcal{C}$ sets the local epoch number to 0 on behalf of each $\mathcal{P}_i$, and it activates $\mathsf{Reshare}$ protocols of APSS on behalf of n parties $\mathcal{P}_1, \ldots, \mathcal{P}_n$ for each epoch number $\tau \in \{0, 1, 2, \cdots, T-1\}$ but initially provides no input to these $\mathsf{Reshare}$ protocols, and subject to the following interactions with $\mathcal{A}$: // Note: w.l.o.g., $\mathcal{C}$ lets $\mathcal{P}_i$ provide input to $\mathsf{Reshare}^{(\tau)}$ immediately (or after any pre-announced fixed delay) upon entering epoch τ; in the initial epoch 0, this makes each $\mathcal{P}_i$ to provide its initial share as input to $\mathsf{Reshare}^{(0\to 1)}$.
   - (Specify per-epoch mobile corruptions) Before the first party enters epoch $\tau \in \{0, 1, 2, \cdots, T\}$, let the adversary $\mathcal{A}$ specify a set of parties $\mathbb{C}^{(\tau)}$ with $|\mathbb{C}^{(\tau)}| \leq f$ and send the set to the challenger $\mathcal{C}$.
   - (Arbitrarily delay message delivery) Whenever a party $\mathcal{P}_i$ would send a message m associated with epoch τ to another party $\mathcal{P}_j$, the message (along with the identities of $\mathcal{P}_i$ and $\mathcal{P}_j$) is first leaked to the adversary. The delivery of m is then suspended until the adversary $\mathcal{A}$ approves the message's delivery.
   - (Misbehave) Throughout, $\mathcal{A}$ can freely access oracle $\mathcal{O}^{\mathsf{corrupt}}$ (described below) to conduct misbehavior.
4. The adversary $\mathcal{A}$ outputs a guessing bit b', and $\mathcal{A}$ wins if $b' = b$.
   // Note: we denote $\mathcal{A}$'s advantage by $\mathsf{Adv}^{G_S^P}(\mathcal{A}) = |\Pr[\text{“}\mathcal{A} \text{ wins”}] - \frac{1}{2}|$.

Corruption oracle $\mathcal{O}^{\mathsf{corrupt}}$ under SINGLE model:

- Initialize $\mathcal{C} \leftarrow \emptyset$ to track the parties under corruption;
- Upon some $\mathcal{P}_i \in \mathbb{C}^{(\tau)}$ enters epoch τ, wait for $\mathcal{A}$'s request to corrupt $\mathcal{P}_i$ while $\mathcal{P}_i$ stays in epoch τ, when for the first time receiving $\mathcal{A}$'s request to corrupt $\mathcal{P}_i$ (at epoch τ), if $|\mathcal{C}| < f$, then $\mathcal{C} \leftarrow \mathcal{C} \cup \{\mathcal{P}_i\}$ and grant the following permissions to $\mathcal{A}$: // Note: here corruption permissions are granted subject to the constraints of SINGLE mobile corruption model as stated in Definition 2
  (i) (partially malicious abilities) if the adversary is *partially* malicious, it can read $\mathcal{P}_i$'s all internal states, and can delete any outgoing message from $\mathcal{P}_i$ associated with epoch τ;
  (ii) (malicious abilities) if the adversary is *fully* malicious, it has partially malicious one's ability, and additionally allows it to:
    a) modify or erase $\mathcal{P}_i$'s local states $s_i^{(\tau)}$ and $\mathsf{aux}^{(\tau)}$;
    b) modify or erase $\mathcal{P}_i$'s incoming messages associated with epoch τ;
    c) send arbitrary messages associated with epoch τ on behalf of $\mathcal{P}_i$;

  until either (i) $\mathcal{A}$ requests to release $\mathcal{P}_i$ from corruption, or (ii) $\mathcal{P}_i$ enters the next epoch $\tau + 1$ or an even newer epoch; upon either event, update $\mathcal{C} \leftarrow \mathcal{C} \setminus \{\mathcal{P}_i\}$ and revoke the above permissions granted to $\mathcal{A}$.
  // Note: if a party moves into a new epoch, the adversary is cleaned, and re-corrupting it needs the adversary to consume a budget from the new epoch; moreover, the adversary can also release a current corruption before the party steps into a new epoch, allowing it to gain a global quota to corrupt some other party (in a different epoch).

This flexibility serves as an asynchronous analogue of synchronous mobile corruption, allowing the adversary to migrate corruptions across different parties currently staying at different local epochs (which, in reality, means some corrupted party in a local epoch τ cleans the adversary and recovers, while at the same time, the adversary compromises some other party in a different local epoch τ'). Following this rationale, our model not only allows a mobile adversary to corrupt another party in a later epoch after releasing a current corruption, but also allows it to corrupt a party that remains "trapped" in an earlier epoch (where the party's local epoch does not advance because messages sent to it are intentionally delayed), after releasing the current corruption.

Correctness Game G_S^C (for APSS under SINGLE model):

The game proceeds as follows between a PPT adversary $\mathcal{A}$ and a challenger $\mathcal{C}$.

1. The adversary $\mathcal{A}$ chooses a secret $s \in \mathbb{F}$ and gives it to the challenger.
2. The challenger runs $(s_1^{(0)}, \ldots, s_n^{(0)}; \mathsf{aux}^{(0)}, \mathsf{pp}) \leftarrow \mathsf{Share}(s)$, and sends $\mathsf{aux}^{(0)}$ and pp to the adversary $\mathcal{A}$.
3. Same as that of G_S^P.
4. The adversary $\mathcal{A}$ instructs the challenger to reconstruct secret in epoch T, namely, After the T $\mathsf{Reshare}$ instances, the challenger activates the $\mathsf{Reconstruct}$ protocol associated with epoch T for all parties. Note that the adversary still has the same permissions as in Step 3.
5. $\mathcal{A}$ wins if any party $\mathcal{P}_i$ that is not corrupted during the epoch T receives a secret s' from the $\mathsf{Reconstruct}$ protocol such that $s' \neq s$.

   // Note: we denote $\mathcal{A}$'s advantage in the game by $\mathsf{Adv}^{G_S^C}(\mathcal{A}) = |\Pr[\text{“}\mathcal{A} \text{ wins”}]|$.

Corruption oracle $\mathcal{O}^{\mathsf{corrupt}}$: same as in game G_S^P.

Liveness Game G_S^L (for APSS under SINGLE model):

The game proceeds as follows between a PPT adversary $\mathcal{A}$ and a challenger $\mathcal{C}$.

1–4. Same as that of G_S^C, except that the adversary $\mathcal{A}$ eventually approves all suspended messages associated with an epoch number $\tau \leq T$ to be delivered (with the exception of any messages deleted using $\mathcal{O}^{\mathsf{corrupt}}$).

5. $\mathcal{A}$ wins if there exists some party $\mathcal{P}_i$ that is not corrupted during the epoch T but does not receive an output from the reconstruction protocol.

   //Note: we denote $\mathcal{A}$'s advantage in the game by $\mathsf{Adv}^{G_S^L}(\mathcal{A}) = |\Pr[\text{“}\mathcal{A} \text{ wins”}]|$.

Corruption oracle $\mathcal{O}^{\mathsf{corrupt}}$: same as in game G_S^P.

4 Generalized Impossibilities

Our main task in this section is to establish comprehensive impossibility results for (n, f, t)-APSS across each of our models of interest. To this end, our impossibility analysis presents generic attackers that can be launched by each considered adversary to break at least one property of APSS. We first focus on these adversaries under the SINGLE accounting model for mobile corruptions (in Subsection 4.1 and 4.2) and then conduct similar analyses under the BOTH model (in Subsection 4.3). Our impossibility results are strong in the sense that generic attackers need not *directly* read messages eavesdropped on the communication channels.[15]

4.1 Impossibility in the SINGLE Model and Partially Malicious Adversary

In preparation for our proof of impossibility under the partially malicious adversary $\mathcal{A}^{\mathsf{part}}$ in the SINGLE model, we recall a few definitional capabilities of $\mathcal{A}^{\mathsf{part}}$.

- If $\mathcal{A}^{\mathsf{part}}$ corrupts $\mathcal{P}_i$ at epoch τ, it learns $s_i^{(\tau)}$, as the syntax states that $\mathcal{P}_i$ enters epoch τ only if $\mathcal{P}_i$ receives the corresponding share $s_i^{(\tau)}$;
- If $\mathcal{A}^{\mathsf{part}}$ corrupts $\mathcal{P}_i$ during its $\mathsf{Reshare}$-phase at epoch τ, it also learns $s_i^{(\tau+1)}$ in addition to $s_i^{(\tau)}$ conditioned on that the $\mathsf{Reshare}$-phase can be completed (i.e., at least $t+1$ parties start to execute it), because the syntax states the $\mathsf{Reshare}$'s output to be $s_i^{(\tau+1)}$;
- If $\mathcal{A}^{\mathsf{part}}$ corrupts $\mathcal{P}_i$ at epoch τ, it can drop all outgoing messages sent by $\mathcal{P}_i$ during this epoch τ;
- Finally, recall that $\mathcal{A}^{\mathsf{part}}$ is a mobile adversary that can rotate the corruption set over parties staying at different epochs.

Next, we present two generic attacks that could be launched by a partially malicious adversary $\mathcal{A}^{\mathsf{part}}$, each of which rules out a different parameter range. The first attack below shows that $\mathcal{A}^{\mathsf{part}}$ can force a liveness violation whenever $t > n - f - 1$.

[15] Looking ahead, our feasibility results are also strong, as they assume an authenticated network in which the adversary can eavesdrop on all network communications.

Privacy Game G_B^P (for APSS under BOTH model):

The game proceeds as follows between a PPT adversary $\mathcal{A}$ and a challenger $\mathcal{C}$.

1. Same as that of G_S^P.
2. Same as that of G_S^P.
3. The challenger $\mathcal{C}$ sets the local epoch number to 0 on behalf of each $\mathcal{P}_i$, and it activates $\mathsf{Reshare}$ protocols of APSS on behalf of n parties $\mathcal{P}_1, \ldots, \mathcal{P}_n$ for each epoch number $\tau \in \{0, 1, 2, \cdots, T-1\}$ but initially provides no input to these $\mathsf{Reshare}$ protocols, and subject to the following interactions with $\mathcal{A}$: // Note: w.l.o.g., $\mathcal{C}$ lets $\mathcal{P}_i$ provide input to $\mathsf{Reshare}^{(\tau)}$ immediately (or after any pre-announced fixed delay) upon entering epoch τ; in the initial epoch 0, this makes each $\mathcal{P}_i$ to provide its initial share as input to $\mathsf{Reshare}^{(0\to 1)}$.
   - (Specify per-epoch mobile corruptions under BOTH model) Before the first party enters epoch $\tau \in \{0, 1, 2, \cdots, T-1\}$, the adversary $\mathcal{A}$ specifies a new set of parties $\mathbb{C}^{(\tau)}$ and another set of parties $\mathbb{C}^{(\tau)}_{\mathsf{reshare}}$, such that $\mathbb{C}^{(\tau)}_{\mathsf{reshare}} \subseteq \mathbb{C}^{(\tau)}$ and $|\mathbb{C}^{(\tau-1)}_{\mathsf{reshare}} \cup \mathbb{C}^{(\tau)}| \leq f$.
   - (Arbitrarily delay message delivery) same as that of G_S^P.
   - (Misbehave) Same as that of G_S^P, except that $\mathcal{O}^{\mathsf{corrupt}}$ is redefined as below.
4. Same as that of G_S^P (i.e., $\mathcal{A}$ outputs a guess b' and it wins if $b' = b$).

Corruption oracle $\mathcal{O}^{\mathsf{corrupt}}$ under BOTH model:

- Initialize $\mathcal{C} \leftarrow \emptyset$ to track the parties under corruption;
- Upon some $\mathcal{P}_i \in \mathbb{C}^{(\tau)}$ entering epoch τ, wait for $\mathcal{A}$'s request to corrupt $\mathcal{P}_i$ while $\mathcal{P}_i$ stays in τ:
  - Upon for the first time receiving $\mathcal{A}$'s request to corrupt $\mathcal{P}_i$ (at epoch τ), if $|\mathcal{C}| < f$ then:
    - If $\mathcal{P}_i \in \mathbb{C}^{(\tau)} \setminus \mathbb{C}^{(\tau)}_{\mathsf{reshare}}$ and $\mathcal{P}_i$'s $\mathsf{handoff} = \mathsf{false}$, $\mathcal{C} \leftarrow \mathcal{C} \cup \{\mathcal{P}_i\}$ and grant the following permissions:
      (i) (partially malicious) if the adversary is *partially* malicious, it can read $\mathcal{P}_i$'s all internal states, and also can delete any outgoing message from $\mathcal{P}_i$ associated with epoch τ;
      (ii) (malicious) if the adversary is *fully* malicious, it has partially malicious one's ability, and additionally allows it to:
      a) modify or erase $\mathcal{P}_i$'s local states $s_i^{(\tau)}$ and $\mathsf{aux}^{(\tau)}$;
      b) modify or erase $\mathcal{P}_i$'s incoming messages associated with epoch τ;
      c) send arbitrary messages associated with epoch τ on behalf of $\mathcal{P}_i$;
      until either (i) $\mathcal{A}$ requests to release $\mathcal{P}_i$ from corruption, or (ii) $\mathcal{P}_i$'s $\mathsf{handoff} = \mathsf{true}$ (i.e., until $\mathcal{P}_i$ starts the $\mathsf{Reshare}$ protocol associated with the epoch τ) or enters a new epoch; upon either event, update $\mathcal{C} \leftarrow \mathcal{C} \setminus \{\mathcal{P}_i\}$ and revoke the above permissions granted to $\mathcal{A}$;
    - If $\mathcal{P}_i \in \mathbb{C}^{(\tau)} \cap \mathbb{C}^{(\tau)}_{\mathsf{reshare}}$, $\mathcal{C} \leftarrow \mathcal{C} \cup \{\mathcal{P}_i\}$, grant the following permissions:
      (i) (partially malicious). Same as the above partially malicious permissions;
      (ii) (malicious). Same as the above malicious permissions;
      until either (i) $\mathcal{A}$ requests to release $\mathcal{P}_i$ from corruption, or (ii) $\mathcal{P}_i$ enters the next epoch $\tau + 1$ or an even newer epoch; upon either event, update $\mathcal{C} \leftarrow \mathcal{C} \setminus \{\mathcal{P}_i\}$ and revoke the above permissions granted to $\mathcal{A}$.

Correctness Game G_B^C (for APSS under BOTH model):

The game proceeds as follows between a PPT adversary $\mathcal{A}$ and a challenger $\mathcal{C}$.

1–2. Same as that of G_S^C.
3. Same as that of G_B^P.
4–5. Same as that of G_S^C, i.e., $\mathcal{A}$ wins if any party that is not corrupted during the epoch T reconstructs some $s' \neq s$.

// Note: we denote $\mathcal{A}$'s advantage in the game by $\mathsf{Adv}^{G_B^C}(\mathcal{A}) = |\Pr[\text{“}\mathcal{A} \text{ wins”}]|$.

Corruption oracle $\mathcal{O}^{\mathsf{corrupt}}$: same as in game G_B^P.

Liveness Game G_B^L (for APSS under BOTH model):

The game proceeds as follows between a PPT adversary $\mathcal{A}$ and a challenger $\mathcal{C}$.

1–4. Same as that of G_B^C, except that the adversary $\mathcal{A}$ eventually approves all suspended messages associated with an epoch number $\tau \leq T$ to be delivered (if these messages are not removed).
5. Same as that of G_S^L (i.e., A wins, if some party that is not corrupted during the epoch T does not reconstruct a value).

// Note: we denote $\mathcal{A}$'s advantage in the game by $\mathsf{Adv}^{G_B^L}(\mathcal{A}) = |\Pr[\text{“}\mathcal{A} \text{ wins”}]|$.

Corruption oracle $\mathcal{O}^{\mathsf{corrupt}}$: same as in game G_B^P.

Lemma 1. *For any* APSS *protocol under the* SINGLE *model against a partially malicious adversary, if* $t > n - f - 1$*, there exists a polynomial-time adversary that can always break the liveness of* APSS*.*

Proof. **Attack.** We prove the statement by showing an attack of a partially malicious adversary $\mathcal{A}^{\mathsf{part}}$ as follows.

1. Specify T, such that Reshare will be executed for T times and reconstruction is launched at epoch T;
2. Do nothing except immediately corrupting f parties upon their entry into epoch $T-1$, and subsequently keep them silent by removing all their outgoing messages associated with epoch $T-1$.

Now, there are only $n - f$ honest parties participating in the resharing protocol at epoch $T-1$. Syntactically, $n-f$ must be at least $t+1$, causing $t \leq n-f-1$. Otherwise (i.e., $t > n-f-1$), there are no sufficient parties to complete the resharing protocol, violating the definitional requirement of APSS. □

Lemma 2. *For any* APSS *protocol under the* SINGLE *model against a partially malicious adversary, if* $t < 3f$*, there exists a polynomial-time adversary that can break either privacy or liveness, conditioned on correctness holding.*

Proof. We prove the statement by the following generic attack that can be launched by a partially malicious adversary $\mathcal{A}^{\mathsf{part}}$, and showing that it always breaks some desired property.

First, the case $t < f$ is immediate: since $t+1 \leq f$, the adversary can corrupt $t+1$ parties holding valid shares in the same epoch and directly reconstruct the secret, thus violating privacy. Hence, it suffices to consider the case $f \leq t < 3f$. Let $f^\star := \min\{f, t+1-f\}$.

Attack. The adversary $\mathcal{A}^{\mathsf{part}}$ proceeds as follows.

1. **Partition and delay.** Partition the parties into a *live set* $\mathcal{L}$ of size $t+1$ and a *delayed set* $\mathcal{D}$ of size $|\mathcal{D}| = n-(t+1) = n-t-1$. The adversary withholds all messages destined to parties in $\mathcal{D}$ so that these parties remain stuck in epochs prior to some chosen epoch τ. Note that the adversary can distribute parties in $\mathcal{D}$ stuck in different epochs, such that every epoch has up to f parties stuck there.
2. **First corruption (epoch τ).** During the resharing $\mathsf{Reshare}^{(\tau \to \tau+1)}$, corrupt an arbitrary subset $\mathcal{L}_1 \subset \mathcal{L}$ of size f. Extract the internal state of every party in $\mathcal{L}_1$ and thereby learn the f shares those parties will hold in epoch $\tau+1$.

3. **Advance and second corruption.** Allow the protocol to advance to epoch $\tau+1$. Then corrupt a disjoint subset $\mathcal{L}_2 \subseteq \mathcal{L} \setminus \mathcal{L}_1$ of size $f^\star$, extract their internal state, and learn the additional $f^\star$ shares of epoch $\tau+1$. Moreover, it removes the outgoing messages of all corrupted parties in $\mathcal{L}_2$.
4. **Deliver.** Release the previously withheld messages so that parties in $\mathcal{D}$ receive the delayed deliveries. Note that the adversary can also selectively corrupt parties in $\mathcal{D}$ in a mobile manner.

Analysis. We first observe the following fact: if a delayed party in $\mathcal{D}$ is allowed by the protocol to obtain its correct secret share for epoch $\tau+1$, then this epoch-$(\tau+1)$ share must be derivable solely from its current local state and from the messages delayed during Steps 1–3 of the attack. To see this is true, recall that by Definition 1, a secret share for epoch $(\tau+1)$ is an output of the Reshare protocol in epoch τ. In the above attack, parties in $\mathcal{L}$ are the only parties that provide inputs to the Reshare protocol in epoch τ. Further considering: after Step 3 in the above attack, $t+1$ parties in $\mathcal{L}$ have erased their epoch-τ secret shares and all randomness used in epoch τ's Reshare protocol, as required for meaningful proactive security; moreover, all parties in $\mathcal{D}$ have not yet entered the epoch τ. Consequently, the delayed parties must rely solely on their current local states together with delayed messages to compute their shares for epoch $\tau+1$.

Observing the above fact, we then consider two possible protocol behaviors upon the delivery of delayed messages and show that, in either case, the adversary can force a violation of liveness or privacy under the stated conditions.

Case I (liveness violation). If the protocol does not permit (at least $f^\star$) parties in $\mathcal{D}$ to advance to epoch $\tau+1$ merely by processing delayed deliveries, then the number of *honest* parties in $\mathcal{L}$ that hold valid epoch-$(\tau+1)$ shares is less than $t+1$. To ensure correctness, only honest parties holding valid shares for epoch $\tau+1$ are allowed to provide inputs in epoch $\tau+1$. As a result, there are eventually at most $(t+1-f^\star)+(f^\star-1)=t$ honest parties with valid shares in epoch $\tau+1$, which is insufficient to preserve correct reconstruction/resharing, thus violating liveness in this case.

Case II (privacy violation). Now suppose the protocol allows at least $f^\star$ parties in $\mathcal{D}$ to obtain valid epoch-$(\tau+1)$ shares by processing delayed messages, as is necessary to restore liveness (as argued in Case I). Let $\mathcal{D}' \subseteq \mathcal{D}$ denote any such set of $f^\star$ parties. Since the adversary controls message scheduling and can read the internal state of a corrupted party, it can corrupt those parties (possibly across older epochs), observe the relevant messages and derive the same shares in epoch-$(\tau+1)$ before releasing the corruptions. In this case, the adversary obtains $|\mathcal{L}_1|+|\mathcal{L}_2|+|\mathcal{D}'| = f+2f^\star$ valid shares for epoch $\tau+1$. This quantity is always at least $t+1$ for $f \le t < 3f$. If $t+1-f \le f$, then $f^\star = t+1-f$, and already $|\mathcal{L}_1|+|\mathcal{L}_2| = f+(t+1-f) = t+1$. Otherwise, $f^\star = f$, and the adversary obtains $f+2f^\star = 3f \ge t+1$. Thus, in either case, $\mathcal{A}^{\mathsf{part}}$ learns at least $t+1$ valid shares , allowing it to fully reconstruct the secret and successfully violate privacy in this case.

Conclusion. Combining the two cases, it is clear that if $t < 3f$, then the partially malicious $\mathcal{A}^{\mathsf{part}}$ can force either a privacy breach or a liveness failure, conditioned on correctness holding. □

Theorem 1 (Impossibility of APSS against partially malicious adversaries under the SINGLE model). *If $t < 3f$ or $t > n-f-1$, then no* APSS *protocol exists under the* SINGLE *model against a partially malicious adversary.*

Proof. It is immediate from Lemma 1 and 2. □

4.2 Impossibility in the SINGLE Model and Malicious Adversary

We now extend our analysis to a malicious adversary $\mathcal{A}^{\mathsf{Byz}}$ in the SINGLE model. To establish the corresponding impossibility result, we consider a restricted variant of a malicious mobile adversary with the following capabilities:

– First, $\mathcal{A}^{\mathsf{Byz}}$ has all (definitional) capabilities of $\mathcal{A}^{\mathsf{part}}$;

– Also, if $\mathcal{A}^{\mathsf{Byz}}$ corrupts $\mathcal{P}_i$ at epoch τ, it can additionally (i) erase $s_i^{(\tau)}$ and $\mathsf{aux}^{(\tau)}$ from $\mathcal{P}_i$'s internal states, (ii) erase all incoming messages of $\mathcal{P}_i$ associated with the epoch τ, and (iii) send arbitrary messages on behalf of $\mathcal{P}_i$ associated with the epoch τ.

Note that by proving impossibility in the presence of a restricted adversary, we obtain a stronger impossibility result. (Meanwhile, our constructions assume the 'full power' malicious adversary as originally defined in our model.) We begin by showing a strengthened impossibility regime, $t > n - 2f - 1$, improving upon the earlier $t > n - f - 1$ regime established in the partially malicious setting.

Lemma 3. *For any* APSS *protocol against a malicious adversary under the* Single *model, if $t > n - 2f - 1$, there exists a polynomial-time adversary that can always break liveness, conditioned on correctness holding.*

Proof. We describe a generic attack by $\mathcal{A}^{\mathsf{Byz}}$ to attack liveness if $t > n - 2f - 1$.

Attack. Initially, the adversary $\mathcal{A}^{\mathsf{Byz}}$ allows all n parties to enter epoch 0, then proceeds as follows to launch the attack.

1. **First corruption and message interception.** At epoch 0, $\mathcal{A}^{\mathsf{Byz}}$ corrupts a set $\mathcal{L}_1$ of size f. The adversary intercepts and drops all messages that are associated with $\mathsf{Reshare}^{(0\to1)}$ and transmitted to parties in $\mathcal{L}_1$. By observing the internal states of parties in $\mathcal{L}_1$ and their incoming messages, $\mathcal{A}^{\mathsf{Byz}}$ computes and sends messages (that are supposed to be sent by these parties), so that these parties' behaviors remain honest.
2. **Advance and second corruption.** Of the parties not in $\mathcal{L}_1$, the adversary allows as many as possible to advance to epoch 1 (note that at most $n-f$ can step into epoch 1). Then, the adversary releases its control of parties in $\mathcal{L}_1$, and corrupts another disjoint set $\mathcal{L}_2$ of f parties that have advanced to epoch 1, and deletes all outgoing messages of parties in $\mathcal{L}_2$.

Analysis. First, consider that at Step 2 of the above attack, the $n-f$ parties that have advanced to epoch 1 have already erased their old shares in epoch 0 and all private states used during the $\mathsf{Reshare}^{(0\to1)}$ protocol (per the requirement of meaningful proactive security). Further, given that all $\mathsf{Reshare}^{(0\to1)}$ messages sent to parties in $\mathcal{L}_1$ were also removed, these parties cannot have correct epoch-1 shares. This means that parties in $\mathcal{L}_1$ do not hold valid epoch-1 shares and cannot provide valid inputs to $\mathsf{Reshare}^{(1\to2)}$ protocol. Also, considering that among $n-f$ parties in epoch 1, there could be f corrupted by the adversary who behave as crashed (as their outgoing messages are totally removed). Thus, $n-2f$ must be at least $t+1$, as otherwise, the number of honest parties with correct shares in epoch 1 is not sufficient to ensure the correctness of resharing/reconstruction, and consequently no parties can advance to epoch 2. □

Lemma 4. *For any* APSS *protocol against a malicious adversary under the* Single *model, if $t < 4f$, there exists a polynomial-time adversary that can break either privacy or liveness, conditioned on correctness holding.*

Proof. We prove the conclusion by showing a generic attack that can be launched by the malicious adversary. The construction of the attack remains the same as in Lemma 2, except that the adversary's corruption strategy against $\mathcal{L}_1$ is augmented with the method used in Lemma 3, that is, the adversary not only learns the shares in $\mathcal{L}_1$, but also intercepts and drops its incoming messages (as well as removing its internal states), so that these parties lose any chance to compute any valid shares for epoch τ+1, even after they are released by the adversary.

Case I (liveness violation). If the protocol forbids (at least) $f+f^\star$ parties in $\mathcal{D}$ from advancing to epoch $\tau+1$, then there is a direct liveness violation, as the honest parties in epoch $\tau+1$ are at most t, which is insufficient to reach the reconstruction/resharing threshold.

Note that compared with Case I in proof for Lemma 2, the difference arises in that liveness is violated there if fewer than f parties obtain valid epoch-$(\tau+1)$ shares(instead of $f+f^\star$), because parties in $\mathcal{L}_1$ still hold valid epoch-$(\tau+1)$ shares there. By contrast,

in this malicious case, the adversary can completely strand the parties in $\mathcal{L}_1$ in epoch τ (because it can cause those parties to lose all $\mathsf{Reshare}^{(\tau\to\tau+1)}$ messages and epoch-τ shares. Consequently, these parties do not hold valid epoch-$(\tau+1)$ shares when reconstruction or the next Reshare is invoked. Thus, compared with the partially malicious case, there are f fewer valid share holders available in epoch $\tau+1$, and at least $f+f^\star$ parties in $\mathcal{D}$ must obtain valid epoch-$(\tau+1)$ shares to restore liveness.

Case II (privacy violation). Otherwise, the attack proceeds identically to the partially malicious case to break privacy, and the adversary at least learns $|\mathcal{L}_1|+|\mathcal{L}_2|+f+f^\star = 2f+2f^\star$ correct shares. This quantity is at least $t+1$ whenever $f \le t < 4f$, sufficient to fully recover the secret and subsequently break privacy.

Conclusion. Combining both analyses, we can see that if $t < 4f$, the malicious adversary $\mathcal{A}^{\mathsf{Byz}}$ can force either a privacy violation or a liveness failure. □

Theorem 2 (Impossibility under the Single Model against Malicious Adversary). *If $t < 4f$ or $t > n-2f-1$, then no* APSS *protocol exists under the Single model against a malicious adversary.*

Proof. It is immediate from Lemma 3 and 4. □

4.3 Adapting Single-Model Impossibilities to the Both Model

We can easily adapt the impossibility result in Subsection 4.1, yielding the following statement on the impossibility of APSS against partially malicious adversaries in the Both model. The analysis largely follows the same arguments, except that the adversary is weakened by no longer corrupting $\mathcal{L}_1$ and therefore learns f fewer epoch-$(\tau+1)$ shares. The resulting impossibility statement in the case is as follows, reflecting this definitionally weaker adversary in the Both model:

Theorem 3 (Impossibility of APSS against partially malicious adversaries under the Both model). *If $t < 2f$ or $t > n-f-1$, then no* APSS *protocol exists under the Both model against a partially malicious adversary.*

Proof. The proof is immediate from the following Lemma 5 and 6. □

Lemma 5. *For any* APSS *protocol under the Both model against a partially malicious adversary, if $t > n-f-1$, there exists a polynomial-time adversary that can always break liveness.*

Proof. The attack only uses the corruption quota within a single epoch, so the Both model introduces no additional constraint. Therefore, the analysis is the same as Lemma 1. □

Lemma 6. *For any* APSS *protocol under the Both model against a partially malicious adversary, if $t < 2f$, there exists a polynomial-time adversary that can break either privacy or liveness, conditioned on correctness holding.*

Proof. The attack follows a similar structure as in Lemma 2, except that: the adversary does not execute the *step2: First corruption (epoch τ)*, where the adversary corrupts f shares during $\mathsf{Reshare}^{(\tau\to\tau+1)}$. Accordingly, in Step 3, the adversary directly corrupts an arbitrary subset $\mathcal{L}_2 \subseteq \mathcal{L}$ of size f after the parties advance to epoch $\tau+1$.

Following the same analysis as in Lemma 2, conditioned on correctness holding, either

(i) valid shares in epoch $\tau+1$ are insufficient to reach the reconstruction threshold and thus violate liveness, or

(ii) to maintain liveness, at least f parties in $\mathcal{D}$ advance to epoch $\tau+1$ with valid shares so that the adversary can learn a total of $2f$ shares in epoch $\tau+1$. If $t < 2f$, then the adversary can accumulate enough valid shares to reconstruct the secret, violating privacy.

This completes the proof. □

Similarly to the above extension of the impossibility result in the partially malicious case, the impossibility in the malicious setting can also be adapted to the Both model from the Single model, yielding the following theorem.

Theorem 4 (Impossibility under the BOTH Model against Malicious Adversary). *If $t < 3f$ or $t > n - 2f - 1$, then no* APSS *protocol exists under the BOTH model against a malicious adversary.*

Proof. The proof is immediate from the following Lemma 7 and 8. □

Lemma 7. *For any* APSS *protocol against a malicious adversary under the BOTH model, if $t > n - 2f - 1$, there exists a polynomial-time adversary that can always break liveness, conditioned on correctness holding.*

Proof. The attack follows a similar structure as in Lemma 3, except that: In the first step, the adversary set the corrupted set $\mathcal{L}_1$ as $\mathbb{C}^{(\tau)} \setminus \mathbb{C}^{(\tau)}_{\mathsf{reshare}}$, i.e., it releases the parties in $\mathcal{L}_1$ before the $\mathsf{Reshare}^{(\tau \to \tau+1)}$ begins. According to the corruption oracle $\mathcal{O}^{\mathsf{corrupt}}$ under the BOTH model, the adversary can delete all messages received by these parties during the $\mathsf{Reshare}^{(\tau \to \tau+1)}$ phase. Consequently, the parties in $\mathcal{L}_1$ fail to obtain valid shares for epoch $\tau + 1$ even after they are released. The remainder of the argument is identical to Lemma 3. □

Lemma 8. *For any* APSS *protocol against a malicious adversary under the BOTH model, if $t < 3f$, there exists a polynomial-time adversary that can break either privacy or liveness, conditioned on correctness holding.*

Proof. The attack follows a similar structure as in Lemma 4, except that the corruption strategy against $\mathcal{L}_1$ is replaced by that of Lemma 7.

Under the BOTH model, for the parties in $\mathcal{L}_1$, the adversary can only delete messages they receive during the $\mathsf{Reshare}^{(\tau \to \tau+1)}$ phase, thereby preventing them obtaining valid shares for epoch $\tau + 1$. However, the adversary also cannot learn the shares of $\mathcal{L}_1$ in epoch $\tau + 1$.

Therefore, conditioned on correctness holding, either

(i) the number of valid shares in epoch $\tau + 1$ is insufficient to reach the reconstruction threshold, which violates liveness, or

(ii) to maintain liveness, at least $f + f^\star$ parties in $\mathcal{D}$ advance to epoch $\tau + 1$ so that the adversary can learn a total of $f + 2f^\star$ shares in epoch $\tau + 1$. Since $f + 2f^\star = f + 2\min\{f, t+1-f\} \geq t+1$. If $t < 3f$, then the adversary can accumulate enough valid shares to reconstruct the secret, violating privacy.

This completes the proof. □

REMARK ON APSS PROTOCOLS WITH UNNECESSARILY t. Throughout this Section, $t + 1$ denotes the participation threshold syntactically required to complete the Reshare and Reconstruct protocols. We do not assume that t must coincide with the exact privacy threshold of the underlying secret-sharing scheme. In particular, an APSS protocol may employ a secret-sharing scheme with a smaller privacy threshold while artificially requiring messages from at least $t + 1$ parties before proceeding with resharing or reconstruction. For example, in resharing/reconstruction, it first asks everyone to multicast a meaningless empty message and then makes each party wait for $t + 1$ such messages before starting the actual resharing/reconstruction.

This distinction does not affect the impossibility results above. In the liveness cases of impossibility proofs, we rely only on the fact that the protocol cannot complete when fewer than $t + 1$ parties are able to provide the required inputs. In the privacy cases, the adversary obtains at least $t+1$ valid shares from the same epoch, which are therefore sufficient to reconstruct the secret even when the underlying privacy threshold is smaller. Hence, artificially increasing the syntactic participation threshold beyond the privacy threshold of the underlying sharing scheme does not circumvent our lower bounds.

5 Showing Tight Feasibility by Constructions

Given the impossibility results in Theorems 1, 2, 3, and 4, we can easily derive the resilience upper bound f of APSS under each considered security model:

Corollary 1 (Resilience upper bound of APSS). *The resilience upper bound f of* APSS *protocols is $\lfloor (n-1)/3 \rfloor$ (resp. $\lfloor (n-1)/4 \rfloor$) in the* Both *(resp.* Single*) model against a partially malicious adversary and $\lfloor (n-1)/5 \rfloor$ (resp. $\lfloor (n-1)/6 \rfloor$) in the* Both *(resp.* Single*) model against a malicious adversary.*

To demonstrate the tightness of the above resilience upper bounds, this section will present two optimal-resilience APSS protocols in the partially malicious and malicious settings, respectively. As we will show, both constructions can be instantiated in either the Single or Both model to achieve optimal resilience, by appropriately choosing the secret-sharing degree t.

5.1 An APSS Protocol against Partially Malicious Adversary

We begin with a very simple APSS construction secure against a partially malicious adversary. Under the Single model, instantiating the parameters at the boundary point $f = \lfloor (n-1)/4 \rfloor$ and $t = 3f$ yields optimal resilience. More generally, the same construction remains secure for any threshold satisfying $3f \leq t \leq n - f - 1$ (analysis of which is deferred to Appendix B.2). Under the Both model, the partially malicious protocol follows the same construction, requiring only an adjustment to parameter $f = \lfloor (n-1)/3 \rfloor$ and $t = 2f$.

Following the syntax in Definition 1, our construction consists of a Share algorithm, a Reshare protocol (Fig. 4), and a Reconstruct protocol (Fig. 5). Note that we will fix a designated participating set $\mathcal{P}_{\mathsf{act}} = \{\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_{t+f+1}\}$ consisting of $t + f + 1$ parties. Only parties in $\mathcal{P}_{\mathsf{act}}$ actively participate in Reshare and Reconstruct protocols and contribute shares during Reconstruct. Other parties behave as offline, but may passively receive the reconstruction messages and output the reconstructed secret. Our construction will use the following primitives (cf. Appendix A for their details):

- Public key encryption (PKE) scheme with standard IND-CPA security;
- An (n^*, t^*)-ACS (asynchronous common subset) protocol—a special multi-valued agreement protocol executed by between t^* and n^* parties (inclusive) in which each party inputs a value and outputs a set of values that is guaranteed to contain the inputs of at least t^* distinct parties.
  Note that we instantiate ACS in the so-called full-information setting [35], which does not require participants to manage any secret states that are vulnerable to the mobile adversary; cf. Appendix A.5 for a detailed discussion.

Partially-malicious-secure Share**.** The algorithm is invoked by a dealer to generate the initial shares of a secret s encoded to a field $\mathbb{F}_p$. It is built from a plain Shamir secret sharing scheme, adapted to execute as follows: (i) samples a random polynomial $f(X) \in \mathbb{F}_p[X]_{\leq t}$ such that with degree up to t and $f(0) = s$, (ii) computes a secret share $s_j^{(0)} \leftarrow f(j)$ for each $j \in [1, t+f+1]$, and sets $s_j^{(0)} \leftarrow \perp$ for each $j \in [t+f+2, n]$, (iii) finally, it exclusively distributes $s_j^{(0)}$ to each $\mathcal{P}_j \in [n]$ at the beginning of an APSS scheme, and also samples and publishes a private-public key pair $(\mathsf{esk}^{\mathsf{def}}, \mathsf{epk}^{\mathsf{def}})$ for PKE as public parameters[16].

Partially-malicious-secure Reshare**.** The Reshare protocol enables parties holding valid shares in epoch τ to refresh them to epoch $\tau + 1$, thus allowing each party $\mathcal{P}_i \in \mathcal{P}_{\mathsf{act}}$ to obtain a refreshed share $s_i^{(\tau+1)}$ and thus enter epoch $\tau + 1$. Additionally, a party may still be in an earlier epoch $\tau' < \tau$ and may not provide their private inputs to this execution. According to the syntax, such parties need not provide an input, but may still participate in the protocol and obtain the output of $\mathsf{Reshare}^{(\tau \to \tau+1)}$. Upon obtaining the output,it directly advances to epoch $\tau + 1$ and securely erases all obsolete local states. As Fig. 4 illustrates, the protocol proceeds as follows:

- *Initialization:* Parties initialize two $(|\mathcal{P}_{\mathsf{act}}|, t+1)$-ACS instances $\mathsf{ACS}_{\mathsf{epk}}$ and $\mathsf{ACS}_{\mathsf{share}}$. Then set their encryption key pair to the common default key pair $(\mathsf{esk}^{\mathsf{def}}, \mathsf{epk}^{\mathsf{def}})$.

[16] In our protocols, the use of such a publicly known PKE key pair is merely for presentation brevity, since encrypting under its public key provides no confidentiality and is therefore equivalent to sending the plaintext directly.

Protocol $\mathsf{Reshare}^{(\tau)}$**, for party** $\mathcal{P}_i \in \mathcal{P}_{\mathsf{act}}$

Private inputs: $s_i^{(\tau)}$
Public inputs: a default key PKE pair $(\mathsf{esk}^{\mathsf{def}}, \mathsf{epk}^{\mathsf{def}})$
Private outputs: $s_i^{(\tau+1)}$

Protocol steps:

// Initialization

- Initialize two full-information ACS instances $(|\mathcal{P}_{\mathsf{act}}|, t+1)$-$\mathsf{ACS}_{\mathsf{epk}}^{(\tau)}$ and $(|\mathcal{P}_{\mathsf{act}}|, t+1)$-$\mathsf{ACS}_{\mathsf{share}}^{(\tau)}$ without providing any input to them.
- Set encryption key pair$(\mathsf{esk}_i, \mathsf{epk}_i)$ as default key pair: $(\mathsf{esk}^{\mathsf{def}}, \mathsf{epk}^{\mathsf{def}})$.

// Ephemeral-PK Publish and Agreement

- Upon receiving the private inputs: ▷ Remark: $\mathcal{P}_i$ does not sample a fresh ephemeral PKE key pair, unless its share $s_i^{(\tau)}$ at epoch τ is provided (i.e., the event to set $\mathsf{handoff} = \mathsf{true}$).
  - Generate a fresh encryption key pair $(\mathsf{esk}_i', \mathsf{epk}_i')$.
  - Input epk_i' to the $(|\mathcal{P}_{\mathsf{act}}|, t+1)$-$\mathsf{ACS}_{\mathsf{epk}}^{(\tau)}$ instance
- Upon receiving T_{epk} from the $(|\mathcal{P}_{\mathsf{act}}|, t+1)$-$\mathsf{ACS}_{\mathsf{epk}}^{(\tau)}$ instance as output
  - For each $j \in \mathcal{P}_{\mathsf{act}}$: if $\mathsf{epk}_j' \in T_{\mathsf{epk}}$, set $\mathsf{epk}_j \leftarrow \mathsf{epk}_j'$; otherwise, set $\mathsf{epk}_j \leftarrow \mathsf{epk}^{\mathsf{def}}$
  - If $\mathsf{epk}_i' \in T_{\mathsf{epk}}$, update $\mathsf{esk}_i \leftarrow \mathsf{esk}_i'$.

// Resharing-Ciphertext Publish and Agreement

- Upon receiving and processing both private inputs and T_{epk}:
  - Sample a t-degree polynomial $\phi_i^{(\tau)}(\cdot)$ with $\phi_i^{(\tau)}(0) = s_i^{(\tau)}$.
  - For each party $\mathcal{P}_j \in \mathcal{P}_{\mathsf{act}}$:
    - Let $[s_i^{(\tau)}]_j \leftarrow \phi_i^{(\tau)}(j)$.
    - Set $c_{i,j}^{(\tau)} \leftarrow \mathsf{Enc}(\mathsf{epk}_j, [s_i^{(\tau)}]_j)$, and erase $[s_i^{(\tau)}]_j$ immediately.
  - Input $c_i = \{c_{i,j}^{(\tau)}\}_{j \in [t+f+1]}$ to the $(|\mathcal{P}_{\mathsf{act}}|, t+1)$-$\mathsf{ACS}_{\mathsf{share}}^{(\tau)}$ instance

// New-Share Interpolation and Secure Erasure

- Wait for receiving T_s as output from $(|\mathcal{P}_{\mathsf{act}}|, t+1)$-$\mathsf{ACS}_{\mathsf{share}}^{(\tau)}$ and also for receiving T_{epk}: ▷ Remark: $\mathcal{P}_i$ does not necessarily provide its private inputs to reach the execution below, because $\mathsf{ACS}_{\mathsf{share}}^{(\tau)}$ and $\mathsf{ACS}_{\mathsf{epk}}^{(\tau)}$ can be completed and therefore return output, if a sufficient number of other parties provide their inputs and execute these ACS protocols.
  - Initialize polynomial $\psi_i(\cdot)$ of degree t.
  - For each item in T_s:
    - Parse the item c_j as: $\{c_{j,k}^{(\tau)}\}_{k \in [t+f+1]}$.
    - Compute $[s_j^{(\tau)}]_i = \mathsf{Dec}(\mathsf{esk}_i, c_{j,i}^{(\tau)})$.
    - Assign $\psi_i(j) \leftarrow [s_j^{(\tau)}]_i$.
  - Interpolate $\psi_i(\cdot)$ from $\{\psi_i(j)\}$.
  - Output $s_i^{(\tau+1)} \leftarrow \psi_i(0)$ after securely erasing old shares $s_i^{(\tau)}$, the ephemeral decryption key esk_i, as well as all other intermediate states used in epoch τ or any smaller epoch. ▷ Remark: the output of this new share is the event advancing to a new epoch.

Fig. 4. $\mathsf{Reshare}$ protocol under a partially malicious adversary.

Protocol $\mathsf{Reconstruction}^{(\tau)}$**, for every party** $\mathcal{P}_i$

Private inputs: $s_i^{(\tau)}$
Public outputs: s

Protocol steps:

- Upon receiving the input, multicast $s_i^{(\tau)}$ to all parties.
- Upon receiving $s_j^{(\tau)}$ from $\mathcal{P}_j$ for the first time, add $s_j^{(\tau)}$ to the share set R.
- Upon R contains at least $t+1$ shares, interpolate the corresponding t-degree polynomial and reconstruct the secret s by evaluating it at 0.

Fig. 5. $\mathsf{Reconstruction}$ protocol under a partially malicious adversary.

- *Ephemeral-PK Publish and Agreement:* Upon receiving its private input, $\mathcal{P}_i$ generates a fresh ephemeral encryption key pair $(\mathsf{esk}_i', \mathsf{epk}_i')$ and proposes epk_i' to an $(|\mathcal{P}_{\mathsf{act}}|, t+1)$-

$\mathsf{ACS}_{\mathsf{epk}}$ instance. Upon completion, all parties obtain the same output T_{epk}. Public keys included in T_{epk} are used for subsequent encryption in this Reshare execution. while the default public key is used for parties whose fresh keys are not included. In addition, $\mathcal{P}_i$ updates its decryption key to esk_i' if its own fresh public key is included in T_{epk}.

- *Resharing-Ciphertext Publish and Agreement:* Each party that has received its private input acts as a dealer and reshares its current share. Specifically, $\mathcal{P}_i$ samples a degree-t polynomial $\phi_i^{(\tau)}(\cdot) \in \mathbb{F}_p[X]_{\leq t}$ satisfying $\phi_i^{(\tau)}(0) = s_i^{(\tau)}$. It then computes encrypted subshares $c_{i,j}^{(\tau)} \leftarrow \mathsf{Enc}(\mathsf{epk}_j, [s_i^{(\tau)}]_j)$ for each j that $\mathcal{P}_j \in \mathcal{P}_{\mathsf{act}}$. The collection $\{c_{i,j}^{(\tau)}\}_{j\in[|\mathcal{P}_{\mathsf{act}}|]}$ is submitted to another $(|\mathcal{P}_{\mathsf{act}}|, t+1)$-$\mathsf{ACS}_{\mathsf{share}}$ instance. The ACS guarantees agreement on a common subset T_s of encrypted subshare sets from $t+1$ dealers.
- *New-Share Interpolation and Secure Erasure:* Upon receiving T_s (and also T_{epk}), for each $\mathsf{transcript}_j \in T_s$, $\mathcal{P}_i \in \mathcal{P}_{\mathsf{act}}$ decrypts the ciphertext $c_{j,i}^{(\tau)}$ to obtain $t+1$ valid subshares. Using these values, it interpolates a degree-t polynomial and derives its refreshed share $s_i^{(\tau+1)} \leftarrow \psi_i(0)$.
  Finally, $\mathcal{P}_i \in \mathcal{P}_{\mathsf{act}}$ securely erases its shares and ephemeral decryption key as well as all other intermediate states used in epoch τ (and earlier epochs).

Partially-malicious-secure Reconstruction**.** As Fig. 5 illustrates, the Reconstruction protocol enables parties to recover the secret s in epoch τ. For each $\mathcal{P}_i \in \mathcal{P}_{\mathsf{act}}$ in epoch τ, it multicasts its share $s_i^{(\tau)}$, and once any party collects $t+1$ shares, it interpolates the polynomial to obtain the secret value s.

Tight feasibility under partially malicious adversaries. The security guarantees of our partially-malicious-secure APSS construction in the Single mobile corruption model give us the following feasibility result:

Theorem 5 (Feasibility of Optimal-Resilience APSS **against Partially Malicious under Single Model).** *There exists a secure (n, f, t)-*APSS *(e.g., APSS in Section 5.1) satisfying Definition 4 for $3f \leq t \leq n-f-1$ against a partially malicious adversary $\mathcal{A}^{\mathsf{part}}$ under the Single model, conditioned on (i) the underlying PKE is IND-CPA secure and (ii) the underlying* ACS *is full-information [35] and $(t+f+1, t+1)$-secure against f corruptions in an asynchronous authenticated network without extra setups.*

Moreover, the same protocol can be reparametrized to be partially-malicious-secure under the Both corruption model, revealing the next feasibility:

Theorem 6 (Feasibility of Optimal-Resilience APSS **against Partially Malicious under Both Model).** *There exists a secure (n, f, t)-*APSS *(e.g., APSS in Section 5.1) satisfying Definition 4 for $2f \leq t \leq n-f-1$ against a partially malicious adversary $\mathcal{A}^{\mathsf{part}}$ under the Both model, conditioned on (i) the underlying PKE scheme is IND-CPA secure and (ii) the underlying* ACS *is full-information [35] and $(t+f+1, t+1)$-secure against f corruptions in an asynchronous authenticated network without extra setups.*

We defer the proof of the above tight feasibility statements (i.e., security analysis of our partially-malicious-secure APSS protocol) to Appendix B.2. A complexity analysis is also given in Appendix B.4.

These conclusions also reveal that a partially-malicious-secure protocol is constructible under the impossible regime of a malicious adversary, demonstrating *a separation that a malicious adversary in APSS is strictly stronger than its partially malicious counterpart.*

5.2 An APSS Protocol against Malicious Adversary

We now describe our construction against a malicious adversary by augmenting our partially-malicious-secure protocol. We target the optimal-resilience point $f = \lfloor (n-1)/6 \rfloor$ and $t = 4f$. More generally, the construction remains secure for any threshold satisfying $4f \leq t \leq n-2f-1$. To counter malicious behavior in Reshare, we introduce a non-interactive public verifiable secret sharing (PVSS) scheme (built atop Pedersen commitment and verifiable PKE), which consists of algorithms $(\mathsf{Deal}, \mathsf{VerifyPVSS}, \mathsf{Deliver})$ and allows parties to publicly verify that

Protocol $\mathsf{Reshare}^{(\tau)}$ **for party** $\mathcal{P}_i \in \mathcal{P}_{\mathsf{act}}$

Private inputs: $s_i^{(\tau)}$, $\hat{s}_i^{(\tau)}$

Public inputs: $\mathsf{cm} \leftarrow \{g^{s_j^{(\tau)}} h^{\hat{s}_j^{(\tau)}}\}_{j\in[|\mathcal{P}_{\mathsf{act}}|]}$, $(\mathsf{esk}^{\mathsf{def}}, \mathsf{epk}^{\mathsf{def}})$

Private outputs: $s_i^{(\tau+1)}$, $\hat{s}_i^{(\tau+1)}$

Public outputs: $\mathsf{cm}' \leftarrow \{g^{s_j^{(\tau+1)}} h^{\hat{s}_j^{(\tau+1)}}\}_{j\in[|\mathcal{P}_{\mathsf{act}}|]}$

Protocol steps:

// Initialization

- Initialize two full-information ACS instances $(|\mathcal{P}_{\mathsf{act}}|, t+1)$-$\mathsf{ACS}_{\mathsf{epk}}^{(\tau)}$ and $(|\mathcal{P}_{\mathsf{act}}|, t+1)$-$\mathsf{ACS}_{\mathsf{share}}^{(\tau)}$ without providing any input to them.
- Set encryption key pair$(\mathsf{esk}_i, \mathsf{epk}_i)$ as default key pair: $(\mathsf{esk}^{\mathsf{def}}, \mathsf{epk}^{\mathsf{def}})$.
- Upon receiving the public inputs, multicast (τ, cm) to all parties.
- Upon receiving $(\tau, \widetilde{\mathsf{cm}})$ with the same commitment $\widetilde{\mathsf{cm}}$ from $2f+1$ distinct parties:
  - If cm^* has not been set in epoch τ, set $\mathsf{cm}^* \leftarrow \widetilde{\mathsf{cm}}$.

// Ephemeral-PK Publish and Agreement

- Upon receiving the private inputs: ▷ Remark: $\mathcal{P}_i$ does not sample a fresh ephemeral PKE key pair, unless its share $s_i^{(\tau)}$ at epoch τ is provided (i.e., the event to set $\mathsf{handoff} = \mathsf{true}$).
  - Generate a fresh encryption key pair $(\mathsf{esk}_i', \mathsf{epk}_i')$.
  - Input epk_i' to an $(|\mathcal{P}_{\mathsf{act}}|, t+1)$-$\mathsf{ACS}_{\mathsf{epk}}^{(\tau)}$
- Upon receiving T_{epk} from the $(|\mathcal{P}_{\mathsf{act}}|, t+1)$-$\mathsf{ACS}_{\mathsf{epk}}^{(\tau)}$ instance as output.
  - For each $j \in \mathcal{P}_{\mathsf{act}}$: if $\mathsf{epk}_j' \in T_{\mathsf{epk}}$, set $\mathsf{epk}_j \leftarrow \mathsf{epk}_j'$; otherwise, set $\mathsf{epk}_j \leftarrow \mathsf{epk}^{\mathsf{def}}$
  - If $\mathsf{epk}_i' \in T_{\mathsf{epk}}$, update $\mathsf{esk}_i \leftarrow \mathsf{esk}_i'$.

// Resharing-Transcript Publish and Agreement

- Upon receiving private input and processing both cm^* and T_{epk}, if $g^{s_i^{(\tau)}} h^{\hat{s}_i^{(\tau)}} = \mathsf{cm}^*[i]$:
  - Invoke $(|\mathcal{P}_{\mathsf{act}}|, t)$-$\mathsf{PVSS.Deal}(s_i^{(\tau)}, \hat{s}_i^{(\tau)}, \{\mathsf{epk}_j\}_{j\in[|\mathcal{P}_{\mathsf{act}}|]})$ to generate a resharing $\mathsf{transcript}_i$.
  - Input $\mathsf{transcript}_i$ to a $(|\mathcal{P}_{\mathsf{act}}|, t+1)$-$\mathsf{ACS}_{\mathsf{share}}^{(\tau)}$.
- Upon obtaining both cm^* and T_{epk}, wait for output T_s, where any element $\mathsf{transcript}_j \in T_s$ must satisfy $\mathsf{Predicate}(\mathsf{transcript}_j, (\mathsf{cm}^*, \{\mathsf{epk}_k\}_{k\in[|\mathcal{P}_{\mathsf{act}}|]}))$.▷ Remark: $\mathcal{P}_i$ does not necessarily provide its private inputs to reach the execution below, because $\mathsf{ACS}_{\mathsf{share}}^{(\tau)}$ and $\mathsf{ACS}_{\mathsf{epk}}^{(\tau)}$ can be completed and therefore return output, if a sufficient number of other parties provide their inputs and execute these ACS protocols.
- Initialize polynomials $\psi_i(\cdot)$ and $\hat{\psi}_i(\cdot)$ of degree t.

// New-Share Interpolation and Secure Erasure

- For each item in T_s:
  - Use $\mathsf{PVSS.Deliver}(\mathsf{transcript}_j, \mathsf{esk}_i)$ to obtain decrypted shares $[s_j^{(\tau)}]_i$, $[\hat{s}_j^{(\tau)}]_i$, and $\{v_{j,k}\}_{k\in[|\mathcal{P}_{\mathsf{act}}|]}$.
  - Assign $\psi_i(j) \leftarrow [s_j^{(\tau)}]_i$ and $\hat{\psi}_i(j) \leftarrow [\hat{s}_j^{(\tau)}]_i$.
- Interpolate $s_i^{(\tau+1)} \leftarrow \psi_i(0)$ and $\hat{s}_i^{(\tau+1)} \leftarrow \hat{\psi}_i(0)$.
- For each $k \in [|\mathcal{P}_{\mathsf{act}}|]$, interpolate $g^{s_k^{(\tau+1)}} h^{\hat{s}_k^{(\tau+1)}}$ using $\{v_{j,k}\}_{\mathsf{transcript}_j \in T_s}$.
- Set $\mathsf{cm}' \leftarrow \{g^{s_k^{(\tau+1)}} h^{\hat{s}_k^{(\tau+1)}}\}_{k\in[|\mathcal{P}_{\mathsf{act}}|]}$.
- Output $(s_i^{(\tau+1)}, \hat{s}_i^{(\tau+1)})$ and cm' after securely erasing old shares $s_i^{(\tau)}$, $\hat{s}_i^{(\tau)}$, decryption key esk_i, as well as all other intermediate states in epoch τ.

Predicate(epk_j) **for** $\mathsf{ACS}_{\mathsf{epk}}$**:**

- Accept if epk_j is a well-formed public key of the underlying PKE scheme; otherwise reject.

Predicate$(\mathsf{transcript}_j, (\mathsf{cm}^*, \{\mathsf{epk}_k\}_{k\in[|\mathcal{P}_{\mathsf{act}}|]}))$ **for** $\mathsf{ACS}_{\mathsf{share}}$**:**

- Check that $\mathsf{PVSS.VerifyPVSS}(\mathsf{transcript}_j, \{\mathsf{epk}_k\}_{k\in[|\mathcal{P}_{\mathsf{act}}|]}) = 1$.
- Parse $\mathsf{transcript}_j$ as $\{v_k, c_k, \hat{c}_k, \pi_k\}_{k\in[|\mathcal{P}_{\mathsf{act}}|]}$.
- Interpolate in the exponent using $\{v_{j,k}\}_{k\in[|\mathcal{P}_{\mathsf{act}}|]}$ to compute v_0. Check that $v_0 = \mathsf{cm}^*[j]$.
- Accept the transcript if the above checks pass; otherwise reject.

Fig. 6. Reshare protocol under a malicious adversary. Grey text is identical to the partially-malicious-secure case, and black text is new to realize malicious security.

the dealer has encrypted all correct shares consistent to a valid degree-t polynomial (cf. Appendix A for deferred details of PVSS).

Moreover, different from the partially malicious protocol, where we let $t+f+1$ parties participate in the resharing protocols, our malicious-secure APSS scheme will fix a larger participating set $\mathcal{P}_{\mathsf{act}} = \{\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_{t+2f+1}\}$, consisting of $t+2f+1$ parties. We also focus on the description under the SINGLE model. The construction under the BOTH model is identical, requiring only minor adjustment to the parameter, by setting $f = \lfloor (n-1)/5 \rfloor$ and $t = 3f$.

Malicious-secure Share**.** The algorithm is invoked by a dealer to compute the initial shares of a secret s. Specifically, the dealer (i) samples random polynomials $f(X) \in \mathbb{F}_p[X]_{\leq t}$ and $\hat{f}(X) \in \mathbb{F}_p[X]_{\leq t}$ of degree up to t and $f(0) = s$, $\hat{f}(0) = \hat{s}$, (ii) computes shares $s_j^{(0)} \leftarrow f(j)$ and $\hat{s}_j^{(0)} \leftarrow \hat{f}(j)$ for each $j \in [1, t+2f+1]$, and sets $s_j^{(0)} \leftarrow \perp$ for each $j \in [t+2f+2, n]$, and (iii) returns $(s_j^{(0)}, \hat{s}_j^{(0)})$ for each $j \in [n]$, which is exclusively sent to $\mathcal{P}_j$ at the beginning of APSS; it also returns a secret-public key pair $(\mathsf{esk}^{\mathsf{def}}, \mathsf{epk}^{\mathsf{def}})$ for PKE as public parameters and the commitment vector $\{g^{s_k^{(0)}} h^{\hat{s}_k^{(0)}}\}_{k \in [|\mathcal{P}_{\mathsf{act}}|]}$ as public auxiliary information.

Protocol $\mathsf{Reconstruction}^{(\tau)}$ **Protocol for every party** $\mathcal{P}_i$

Private inputs: $s_i^{(\tau)}$ and $\hat{s}_i^{(\tau)}$
Public inputs: $\mathsf{cm} = \{g^{s_j^{(\tau)}} h^{\hat{s}_j^{(\tau)}}\}_{j \in [|\mathcal{P}_{\mathsf{act}}|]}$
Public outputs: s

Protocol steps:

- Upon receiving the public inputs, multicast (τ, cm) to all parties.
- Upon receiving $(\tau, \widetilde{\mathsf{cm}})$ with the same commitment $\widetilde{\mathsf{cm}}$ from $2f+1$ distinct parties:
  - If cm^* has not been set, set $\mathsf{cm}^* \leftarrow \widetilde{\mathsf{cm}}$.
- Upon receiving the private input, multicast $(s_i^{(\tau)}, \hat{s}_i^{(\tau)})$ to all parties.
- Upon receiving $(s_j^{(\tau)}, \hat{s}_j^{(\tau)})$ from $\mathcal{P}_j$:
  - wait until cm^* has been set in epoch τ:
    - If $g^{s_j^{(\tau)}} h^{\hat{s}_j^{(\tau)}} = \mathsf{cm}^*[j]$, add $s_j^{(\tau)}$ to the valid share set R.
- Upon R contains at least $t+1$ valid shares, interpolate the corresponding t-degree polynomial and reconstruct the secret s by evaluating it at 0.

Fig. 7. $\mathsf{Reconstruction}$ protocol under a malicious adversary. Grey text is identical to the partially-malicious-secure case, and black text is new to realize malicious security.

Malicious-secure $\mathsf{Reshare}$**.** Each party $\mathcal{P}_i \in \mathcal{P}_{\mathsf{act}}$ initially holds a pair of shares $(s_i^{(\tau)}, \hat{s}_i^{(\tau)})$ in epoch τ. The protocol outputs fresh shares $(s_i^{(\tau+1)}, \hat{s}_i^{(\tau+1)})$ together with updated commitments to the new sharing. The procedure proceeds in three main steps:

- *Initialization.* Identical to that in the partially malicious construction, except that $|\mathcal{P}_{\mathsf{act}}| = t+2f+1$. Parties also multicast their local commitment vectors and fix cm^* upon receiving the same vector from $2f+1$ distinct parties.
- *Ephemeral-PK Publish and Agreement.* The key-update phase is identical to that in the partially malicious construction, except that $|\mathcal{P}_{\mathsf{act}}| = t+2f+1$.
- *Resharing-Transcript Publish and Agreement.* After obtaining the fixed commitment cm^* and the agreed ephemeral public keys T_{epk}, each party that has received its private input first checks whether its private shares are consistent with cm^*. If so, it acts as a dealer to generate a PVSS resharing transcript $\mathsf{transcript}_i$ and proposes the transcript to $\mathsf{ACS}_{\mathsf{share}}$. The $\mathsf{ACS}_{\mathsf{share}}$ employs an external validity predicate to ensure that only valid resharing transcripts consistent with the fixed commitments are included in the common output T_s of size $t+1$
- *New-Share Interpolation and Secure Erasure.* Upon receiving the outputs T_s, $\mathcal{P}_i \in \mathcal{P}_{\mathsf{act}}$ invokes the $\mathsf{PVSS.Deliver}$ algorithm to retrieve the sub-shares $[s_j^{(\tau)}]_i$ and $[\hat{s}_j^{(\tau)}]_i$ from each $\mathsf{transcript}_j \in T_s$, along with the associated commitments $\{v_{j,k}\}_{k \in [|\mathcal{P}_{\mathsf{act}}|]}$. $\mathcal{P}_i$ defines two polynomials, $\psi_i(x)$ and $\hat{\psi}_i(x)$, such that for each $j \in T_s$, $\psi_i(j) = [s_j^{(\tau)}]_i$ and $\hat{\psi}_i(j) = [\hat{s}_j^{(\tau)}]_i$. By evaluating these polynomials at 0, $\mathcal{P}_i$ computes its new shares.

Concurrently, $\mathcal{P}_i$ interpolates the new public commitments cm' for all parties.

Finally, to ensure forward security, $\mathcal{P}_i$ securely erases its old shares $s_i^{(\tau)}$, $\hat{s}_i^{(\tau)}$, the old private key esk_i as well as all other intermediate states in epoch τ.

Malicious-secure Reconstruction**.** This largely follows the partially malicious version, except that each party additionally multicasts its local commitment vector and blinding share $\hat{s}_i^{(\tau)}$. Upon receiving the same commitment vector from $2f+1$ distinct parties, a party fixes it as cm^*, and verifies each revealed share against cm^* before accepting it for interpolation.

Tight feasibility under malicious adversaries. The following feasibility can be derived from our malicious-secure APSS protocol in the SINGLE model:

Theorem 7 (Feasibility of Optimal-Resilience APSS **against Malicious Adversary under SINGLE Model).** *There exists a secure (n, f, t)-*APSS *(e.g., APSS in Section 5.2) satisfying Definition 4 for $4f \leq t \leq n - 2f - 1$ against a malicious adversary $\mathcal{A}^{\mathsf{Byz}}$ under the SINGLE model, conditioned on (i) the hardness of discrete logarithm problem, (ii) the underlying verifiable PKE scheme satisfies completeness, soundness, and zero-knowledge, and (iii) underlying* ACS *is full-information [35] and $(t + 2f + 1, t + 1)$-secure against f malicious corruptions in an asynchronous authenticated network without extra setups.*

The same maliciously secure protocol can also be parametrized against the BOTH mobile corruption model, leading to a feasibility summarized below:

Theorem 8 (Feasibility of Optimal-Resilience APSS **against Malicious Adversary under BOTH Model).** *There exists a secure (n, f, t)-*APSS *(e.g.,* APSS *in Section 5.2) satisfying Definition 4 for $3f \leq t \leq n - 2f - 1$ against a malicious adversary $\mathcal{A}^{\mathsf{Byz}}$ under the BOTH model, conditioned on (i) the hardness of discrete logarithm problem, (ii) the underlying verifiable PKE scheme satisfies completeness, soundness, and zero-knowledge, and (iii) underlying* ACS *is full-information [35] and $(t + 2f + 1, t + 1)$-secure against f malicious corruptions in an asynchronous authenticated network without extra setups.*

We defer the proof of the above feasibility theorems (i.e., security analysis of our malicious-secure APSS protocol) to Appendix B.3. A complexity analysis is also given in Appendix B.4.

5.3 Separation between SINGLE and BOTH Models

Beyond the separation that malicious adversaries are strictly stronger than partially malicious ones (due to the gap between feasibility in the partially malicious setting and impossibility in the malicious setting), we also establish a separation between the SINGLE and BOTH models via constructions in the section.

Corollary 2. *The SINGLE model captures a strictly stronger adversary than the BOTH model, regardless of whether the adversary is partially or fully malicious.*

Proof. We establish this separation by showing constructions secure in the BOTH model but impossible in the SINGLE model under the same regimes of (n, f, t).

First, consider the case under a partially malicious adversary. By Theorem 6, the APSS protocol is secure if $2f \leq t \leq n - f - 1$ against a partially malicious adversary under the BOTH model. For example, setting $n = 3f + 1$ and $t = 2f$ satisfies this condition. On the other hand, Theorem 1 shows that under the SINGLE model, no APSS protocol exists if $t < 3f$ or $t > n - f - 1$. Hence, for the same parameters $n = 3f + 1$ and $t = 2f$, the protocol is secure under the BOTH model but impossible under the SINGLE model, yielding a strict separation.

The same reasoning applies to the case of a malicious adversary: $n = 5f + 1$ and $t = 3f$ correspond to a parameter choice that is secure under the BOTH model but impossible under the SINGLE model, thereby establishing the separation in the malicious setting as well.

References

1. Abraham, I., Malkhi, D., Spiegelman, A.: Asymptotically optimal validated asynchronous byzantine agreement. In: Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing. pp. 337–346 (2019)
2. Alexandru, A.B., Blum, E., Katz, J., Loss, J.: State machine replication under changing network conditions. In: Advances in Cryptology–ASIACRYPT 2022. pp. 681–710
3. Almansa, J.F., Damgård, I., Nielsen, J.B.: Simplified threshold rsa with adaptive and proactive security. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 593–611. Springer (2006)
4. Alon, N., Galil, Z., Yung, M.: Efficient dynamic-resharing "verifiable secret sharing" against mobile adversary. In: European Symposium on Algorithms. pp. 523–537. Springer (1995)
5. Backes, M., Kate, A., Patra, A.: Computational verifiable secret sharing revisited. In: International Conference on the Theory and Application of Cryptology and Information Security. pp. 590–609. Springer (2011)
6. Bandarupalli, A., Ji, X., Jog, S., Kate, A., Liu-Zhang, C.D., Song, Y.: Computationally and communication efficient batched asynchronous dpss from lightweight cryptography. Cryptology ePrint Archive (2025)
7. Baron, J., Defrawy, K.E., Lampkins, J., Ostrovsky, R.: Communication-optimal proactive secret sharing for dynamic groups. In: International conference on applied cryptography and network security. pp. 23–41. Springer (2015)
8. Baron, J., El Defrawy, K., Lampkins, J., Ostrovsky, R.: How to withstand mobile virus attacks, revisited. In: Proceedings of the 2014 ACM symposium on Principles of distributed computing. pp. 293–302 (2014)
9. Ben-Or, M., Canetti, R., Goldreich, O.: Asynchronous secure computation. In: Proceedings of the twenty-fifth annual ACM symposium on Theory of computing. pp. 52–61 (1993)
10. Ben-Or, M., Goldwasser, S., Wigderson, A.: Completeness theorems for non-cryptographic fault-tolerant distributed computation. In: Proceedings of the twentieth annual ACM symposium on Theory of computing. pp. 1–10 (1988)
11. Ben-Or, M., Kelmer, B., Rabin, T.: Asynchronous secure computations with optimal resilience. In: Proceedings of the thirteenth annual ACM symposium on Principles of distributed computing. pp. 183–192 (1994)
12. Benhamouda, F., Gentry, C., Gorbunov, S., Halevi, S., Krawczyk, H., Lin, C., Rabin, T., Reyzin, L.: Can a public blockchain keep a secret? In: Theory of Cryptography Conference. pp. 260–290. Springer (2020)
13. Bracha, G.: Asynchronous byzantine agreement protocols. Information and computation **75**(2), 130–143 (1987)
14. Cachin, C., Kursawe, K., Lysyanskaya, A., Strobl, R.: Asynchronous verifiable secret sharing and proactive cryptosystems. In: Proceedings of the 9th ACM Conference on Computer and Communications Security. pp. 88–97 (2002)
15. Cachin, C., Kursawe, K., Petzold, F., Shoup, V.: Secure and efficient asynchronous broadcast protocols. In: Annual International Cryptology Conference. pp. 524–541. Springer (2001)
16. Cachin, C., Kursawe, K., Shoup, V.: Random oracles in constantipole: practical asynchronous byzantine agreement using cryptography. In: Proceedings of the nineteenth annual ACM symposium on Principles of distributed computing. pp. 123–132 (2000)
17. Canetti, R., Gennaro, R., Goldfeder, S., Makriyannis, N., Peled, U.: UC non-interactive, proactive, threshold ECDSA with identifiable aborts. In: Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security. pp. 1769–1787 (2020)
18. Canetti, R., Gennaro, R., Jarecki, S., Krawczyk, H., Rabin, T.: Adaptive security for threshold cryptosystems. In: Annual International Cryptology Conference. pp. 98–116. Springer (1999)
19. Canetti, R., Rabin, T.: Fast asynchronous byzantine agreement with optimal resilience. In: Proceedings of the twenty-fifth annual ACM symposium on Theory of computing. pp. 42–51 (1993)
20. Cascudo, I., David, B.: Publicly verifiable secret sharing over class groups and applications to dkg and yoso. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 216–248. Springer (2024)
21. Cascudo, I., David, B., Garms, L., Konring, A.: Yolo yoso: fast and simple encryption and secret sharing in the yoso model. In: International Conference on the Theory and Application of Cryptology and Information Security. pp. 651–680. Springer (2022)
22. Damgård, I.B., Kamp, S.H., Loss, J., Nielsen, J.B.: Asynchronous YOSO a la paillier. Cryptology ePrint Archive, Paper 2025/128 (2025), `https://eprint.iacr.org/2025/128`
23. Das, S., Xiang, Z., Tomescu, A., Spiegelman, A., Pinkas, B., Ren, L.: Verifiable secret sharing simplified. In: 2025 IEEE Symposium on Security and Privacy (SP). pp. 633–651. IEEE (2025)

24. Desmedt, Y., Frankel, Y.: Shared generation of authenticators and signatures. In: Annual International Cryptology Conference. pp. 457–469. Springer (1991)
25. Dolev, S., Eldefrawy, K., Lampkins, J., Ostrovsky, R., Yung, M.: Proactive secret sharing with a dishonest majority. In: International Conference on Security and Cryptography for Networks. pp. 529–548. Springer (2016)
26. Eldefrawy, K., Genise, N., Kshirsagar, R., Yung, M.: On regenerating codes and proactive secret sharing: Relationships and implications. In: International symposium on stabilizing, safety, and security of distributed systems. pp. 350–364. Springer International Publishing Cham (2021)
27. Feldman, P.: A practical scheme for non-interactive verifiable secret sharing. In: 28th Annual Symposium on Foundations of Computer Science (sfcs 1987). pp. 427–438. IEEE (1987)
28. Feng, H., Gao, Y., Lu, Y., Tang, Q., Xu, J.: Practical asynchronous distributed key reconfiguration and its applications. Cryptology ePrint Archive (2025)
29. Frankel, Y., Gemmell, P., MacKenzie, P.D., Yung, M.: Optimal-resilience proactive public-key cryptosystems. In: Proceedings 38th Annual Symposium on Foundations of Computer Science. pp. 384–393. IEEE (1997)
30. Gentry, C., Halevi, S., Krawczyk, H., Magri, B., Nielsen, J.B., Rabin, T., Yakoubov, S.: YOSO: You only speak once: Secure mpc with stateless ephemeral roles. In: Annual International Cryptology Conference. pp. 64–93. Springer (2021)
31. Günther, C.U., Das, S., Kokoris-Kogias, L.: Practical asynchronous proactive secret sharing and key refresh. Cryptology ePrint Archive (2022)
32. Herzberg, A., Jakobsson, M., Jarecki, S., Krawczyk, H., Yung, M.: Proactive public key and signature systems. In: Proceedings of the 4th ACM Conference on Computer and Communications Security. pp. 100–110 (1997)
33. Herzberg, A., Jarecki, S., Krawczyk, H., Yung, M.: Proactive secret sharing or: How to cope with perpetual leakage. In: annual international cryptology conference. pp. 339–352. Springer (1995)
34. Hu, B., Liu, J., Lu, Z., Tang, Q., Xiang, Z., Zhang, Z.: Optimistic asynchronous dynamic-committee proactive secret sharing. Cryptology ePrint Archive (2025)
35. Huang, S.E., Pettie, S., Zhu, L.: Byzantine agreement with optimal resilience via statistical fraud detection. Journal of the ACM **71**(2), 1–37 (2024)
36. Kondi, Y., Magri, B., Orlandi, C., Shlomovits, O.: Refresh when you wake up: Proactive threshold wallets with offline devices. In: 2021 IEEE Symposium on Security and Privacy (SP). pp. 608–625. IEEE (2021)
37. Levrat, C., Rambaud, M., Urban, A.: Breaking the $t < n/3$ consensus bound: Asynchronous dynamic proactive secret sharing under honest majority. Cryptology ePrint Archive (2022)
38. Maram, S.K.D., Zhang, F., Wang, L., Low, A., Zhang, Y., Juels, A., Song, D.: Churp: dynamic-committee proactive secret sharing. In: Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security. pp. 2369–2386 (2019)
39. Nikov, V., Nikova, S.: On proactive secret sharing schemes. In: International Workshop on Selected Areas in Cryptography. pp. 308–325. Springer (2004)
40. Ostrovsky, R., Yung, M.: How to withstand mobile virus attacks. In: Proceedings of the tenth annual ACM symposium on Principles of distributed computing. pp. 51–59 (1991)
41. Paillier, P.: Public-key cryptosystems based on composite degree residuosity classes. In: International conference on the theory and applications of cryptographic techniques. pp. 223–238. Springer (1999)
42. Patra, A., Choudhary, A., Pandu Rangan, C.: Simple and efficient asynchronous byzantine agreement with optimal resilience. In: Proceedings of the 28th ACM symposium on Principles of distributed computing. pp. 92–101 (2009)
43. Pedersen, T.P.: Non-interactive and information-theoretic secure verifiable secret sharing. In: Annual international cryptology conference. pp. 129–140. Springer (1991)
44. Schultz, D., Liskov, B., Liskov, M.: Mpss: mobile proactive secret sharing. ACM Transactions on Information and System Security (TISSEC) **13**(4), 1–32 (2010)
45. Shamir, A.: How to share a secret. Communications of the ACM **22**(11), 612–613 (1979)
46. Xing, X., Liu, Y., Liao, B., Liu, J., Hu, B., Lin, X., Lu, Y., Zhang, T.: Gossamer: Lightweight and linear-communication asynchronous (dynamic proactive) secret sharing and the applications. Cryptology ePrint Archive (2025)
47. Yan, Y., Xia, Y., Devadas, S.: Shanrang: Fully asynchronous proactive secret sharing with dynamic committees. Cryptology ePrint Archive (2022)
48. Yurek, T., Xiang, Z., Xia, Y., Miller, A.: Long live the honey badger: Robust asynchronous DPSS and its applications. In: 32nd USENIX Security Symposium (USENIX Security 23). pp. 5413–5430 (2023)

A Construction Ingredients

Below, we provide more details about the primitives used in our APSS constructions.

A.1 Pedersen Commitment

Given a cyclic group $\mathbb{G}_p$ with prime order p and its two random generators g and h as public parameters, Pedersen commitment (PC) [43] is a tuple of two algorithms as follows:

- $\mathsf{Commit}(m, r; pp) \rightarrow v$. On input message m and a randomness r, return the commitment $v = g^m h^r$, where pp are the public parameters of g and h.
- $\mathsf{VerifyCom}(v, m', r'; pp) \rightarrow \{0, 1\}$. On input the commitment v and possible opening of it m' and r', return 1 if $v = g^{m'} h^{r'}$, otherwise return 0.

PC is correct, as $\mathsf{VerifyCom}(\mathsf{com}, m, r)$ always returns 1, if $\mathsf{com} \leftarrow \mathsf{Commit}(m; r)$. PC is information-theoretic hiding, and is also computationally binding conditioned on the hardness of the discrete logarithm problem.

A.2 Public-Key Encryption

A public-key encryption (PKE) scheme consists of a tuple of three polynomial-time algorithms $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$:

- $\mathsf{KeyGen}(1^\lambda) \rightarrow (\mathsf{epk}, \mathsf{esk})$: generate a public-secret key pair.
- $\mathsf{Enc}(\mathsf{epk}, m) \rightarrow c$: encrypt a message m under the public key epk, and output a ciphertext c.
- $\mathsf{Dec}(\mathsf{esk}, c) \rightarrow m$ or $\perp$: decrypt a ciphertext c using the secret key esk to output a plaintext m (or $\perp$ if decryption fails).

A PKE scheme is correct if for any key pair $(\mathsf{epk}, \mathsf{esk})$ output by KeyGen and any message m in the message space, $\mathsf{Dec}(\mathsf{esk}, \mathsf{Enc}(\mathsf{epk}, m)) = m$ holds except with negligible probability. Throughout the paper, we also assume the standard indistinguishability under chosen-plaintext attack (IND-CPA) security for PKE unless stated otherwise.

A.3 Verifiable PKE for Opening of Pedersen Commitment

In our maliciously secure APSS construction, we additionally require the underlying PKE scheme to support auxiliary interfaces that enable zero-knowledge proofs attesting to the fact that they are encrypting the correct opening to a corresponding Pedersen commitment. In particular, we require two additional algorithms, corresponding to a zero-knowledge proof scheme as follows:

- $\mathsf{EncAndProve}(m, r; \mathsf{epk}) \rightarrow (c, \hat{c}, \pi)$. Given the opening m and r to a Pedersen commitment $v = g^m h^r$, compute two PKE ciphertexts $c \leftarrow \mathsf{Enc}(\mathsf{epk}, m)$ and $\hat{c} \leftarrow \mathsf{Enc}(\mathsf{epk}, r)$, and generate a zero-knowledge proof π for the statement $\{(m, r) : v = g^m h^r \ \wedge \ c = \mathsf{Enc}(\mathsf{epk}, m) \ \wedge \ \hat{c} = \mathsf{Enc}(\mathsf{epk}, r)\}$, i.e., the two values encrypted under epk are correct opening to the commitment v.
- $\mathsf{Verify}(\mathsf{epk}, v, c, \hat{c}, \pi) \rightarrow \{0, 1\}$. Return 1 (resp.0) if accept (resp. reject) the statement.

The above scheme can be built atop the Paillier encryption scheme [41,48] or the exponential ElGamal encryption [23], and shall satisfy the properties of completeness, soundness, and zero-knowledge.

- *Completeness.* For any given public parameters and encryption key, the $\mathsf{EncAndProve}$ algorithm can always return $(c, \hat{c}, \pi)$, such that $\mathsf{Verify}(\mathsf{epk}, v = g^m h^r, c, \hat{c}, \pi) = 1$, for any m and $r \in \mathbb{Z}_p$.
- *Soundness.* Any PPT adversary cannot provide a valid proof if c and $\hat{c}$ are encrypting m' and r' such that $g^{m'} h^{r'} \neq v$.

– *Zero-knowledge.* There exists a probabilistic polynomial-time simulator $\mathcal{S}$ such that for all public encryption keys epk, and for all tuples $(v, c, \hat{c})$ satisfying

$$v = g^m h^r, \quad c = \mathsf{Enc}(\mathsf{epk}, m), \quad \hat{c} = \mathsf{Enc}(\mathsf{epk}, r),$$

the simulator $\mathcal{S}$, on input $(\mathsf{epk}, v, c, \hat{c})$ and given access to the setup trapdoor td, but without knowing m or r, outputs a simulated proof π^{sim} that is computationally indistinguishable from π that is actually computed by $\mathsf{EncAndProve}(m, r; \mathsf{epk}) \to (c, \hat{c}, \pi)$.

A.4 Publicly Verifiable Secret Sharing

Below, we describe a non-interactive (n, t) publicly verifiable secret sharing (PVSS) scheme built atop Pedersen polynomial commitment and verifiable PKE. Given public parameters, this PVSS scheme can be abstracted as a tuple of three algorithms $(\mathsf{Deal}, \mathsf{VerifyPVSS}, \mathsf{Deliver})$, where Deal is executed by a dealing party to compute a sharing transcript to distribute shares among n recipients, $\mathsf{VerifyPVSS}$ can be run by anyone to verify whether the sharing transcript is well-formed as encrypting n shares consistent with a fixed t-degree polynomial or not, and the $\mathsf{Deliver}$ algorithm is run by a recipient to get its corresponding secret share from the sharing transcript. Such a non-interactive (n, t) PVSS scheme consisting of $(\mathsf{Deal}, \mathsf{VerifyPVSS}, \mathsf{Deliver})$ algorithms can be constructed as follows [48,23]:

– $\mathsf{Deal}(s, \hat{s}, \{\mathsf{epk}_j\}_{j\in[n]}) \to \mathsf{transcript}$**.** On input the secret s, a random blinding value $\hat{s}$, and the public encryption keys $\{\mathsf{epk}_j\}_{j\in[n]}$ for all parties, the dealer proceeds as follows:
  - The dealer samples two random polynomials $\phi(\cdot)$ and $\hat{\phi}(\cdot)$ of degree at most t, such that $s \leftarrow \phi(0)$ and $\hat{s} \leftarrow \hat{\phi}(0)$.
  - For each party $\mathcal{P}_j$, the dealer defines the secret share pair $(sh_j, \hat{sh}_j) \leftarrow (\phi(j), \hat{\phi}(j))$. The dealer then computes a Pedersen commitment for each pair of share points, i.e., $v_j \leftarrow g^{\phi(j)} h^{\hat{\phi}(j)}$.
  - For $j \in [n]$, the dealer computes $\mathsf{EncAndProve}(sh_j, \hat{sh}_j; \mathsf{epk}_j) \to (c_j, \hat{c}_j, \pi_j)$, where c_j and $\hat{c}_j$ are ciphertexts respectively encrypting sh_j and $\hat{sh}_j$ under $\mathcal{P}_j$'s public encryption key epk_j, and π_j is a non-interactive zero-knowledge (NIZK) proof attesting the statement $\{(sh_j, \hat{sh}_j) : v_j = g^{sh_j} h^{\hat{sh}_j} \ \wedge \ c_j = \mathsf{Enc}(\mathsf{epk}_j, sh_j) \ \wedge \ \hat{c}_j = \mathsf{Enc}(\mathsf{epk}_j, \hat{sh}_j)\}$.
  - The dealer finally outputs a $\mathsf{transcript}$ consisting of:
    * Pedersen commitment $v_j = g^{sh_j} h^{\hat{sh}_j}$ for each $j \in [n]$,
    * ciphertexts c_j and $\hat{c}_j$ respectively encrypting $sh_j, \hat{sh}_j$ for each $j \in [n]$,
    * for each $j \in [n]$, a NIZK proof π_j to attest: $\{(sh_j, \hat{sh}_j) : v_j = g^{sh_j} h^{\hat{sh}_j} \ \wedge \ c_j = \mathsf{Enc}(\mathsf{epk}_j, sh_j) \ \wedge \ \hat{c}_j = \mathsf{Enc}(\mathsf{epk}_j, \hat{sh}_j)\}$.

– $\mathsf{VerifyPVSS}(\mathsf{transcript}, \{\mathsf{epk}_j\}_{j\in[n]}) \to 0/1$**:** Given a PVSS $\mathsf{transcript}$ and the public encryption keys, anyone can execute as follows to verify whether it encrypts shares consistent at a t-degree polynomial or not:
  - Parse $\mathsf{transcript}$ as $\{v_j\}_{j\in[n]}$, $\{c_j\}_{j\in[n]}$, $\{\hat{c}_j\}_{j\in[n]}$, and $\{\pi_j\}_{j\in[n]}$.
  - Verify $\{v_j\}_{j\in[n]}$ lie on a common polynomial with degree at most t.
  - Verify the correctness of the NIZK proof of verifiable PKE for Pedersen commitment, i.e. $\mathsf{Verify}(\mathsf{epk}_j, v_j, c_j, \hat{c}_j, \pi_j) = 1$, for each $j \in [n]$.
  - If any of the above checks fail, output 0. Otherwise, output 1.

– $\mathsf{Deliver}(\mathsf{transcript}, \mathsf{esk}_i) \to (sh_i, \hat{sh}_i, \mathsf{aux}_i)$**.** Given the sharing $\mathsf{transcript}$, a recipient $\mathcal{P}_i$ having the decryption key esk_i, executes as follows to compute its secret shares along with auxiliary output:
  - Parse $\mathsf{transcript}$ as $\{v_j\}_{j\in[n]}$, $\{c_j\}_{j\in[n]}$, $\{\hat{c}_j\}_{j\in[n]}$, and $\{\pi_j\}_{j\in[n]}$.
  - Decrypt to get shares $sh_i \leftarrow \mathsf{Dec}(\mathsf{esk}_i, c_i)$ and $\hat{sh}_i \leftarrow \mathsf{Dec}(\mathsf{esk}_i, \hat{c}_i)$.
  - Output $(sh_i, \hat{sh}_i, \{v_j\}_{j\in[n]})$.

A.5 Asynchronous Common Subset

A (n^*, t^*) asynchronous common subset (ACS) protocol is executed among up to n^* designated parties (and at least t^* parties). Each party that remains honest throughout the execution of the protocol can provide as input a proposed value and eventually output a multi-set of inputs proposed by distinct parties, while corrupted parties may behave arbitrarily within the protocol. Such an ACS protocol is said (n^*, t^*)-secure, if it satisfies the following properties, except with negligible probability:

- **Termination.** If at least t^* honest parties propose inputs to an ACS protocol instance, then every honest party will eventually obtain an output from the ACS instance, provided that all messages exchanged among honest parties for that instance are eventually delivered.
- **Agreement.** If an honest party outputs a multi-set T, then all honest parties will output T.
- **Subset Validity.** If an honest party outputs a multi-set T, then $|T| = t^*$ and each element in T is proposed by a distinct party.
- **External Validity.** If an honest party outputs a multi-set T, then for each element in T, it satisfies a global predicate (pre-defined according to public validity conditions).

Instantiations. In our setting, we adopt a polynomial-time ACS instantiation that relies solely on authenticated point-to-point channels, without additional setup assumptions such as a public-key infrastructure or threshold cryptosystems. This avoids requiring parties to maintain private internal secrets (e.g., private signing keys or communication session keys) within each ACS instance, which is particularly problematic against mobile adversaries, as any party's secrets may be exposed across previous or subsequent epochs (even if the party remains honest in the current epoch). By contrast, an ACS instantiation without initial secrets preserves security across a sequence of ACS instances assigned to different epochs, even under asynchronous mobile corruptions, as long as authenticated channels are available.

Following the classic BKR framework [11], such desired polynomial-time ACS protocols with optimal $\lfloor (n-1)/3 \rfloor$ resilience can be built from polynomial-time asynchronous binary Byzantine agreement in the full-information setting due to a recent breakthrough from Huang et al. [35][17]. Recall that, in this full-information setting, the setup assumptions are restricted to the standard authenticated peer-to-peer network, without additional assumptions (e.g., PKI, threshold cryptosystems, pre-shared common coins, or private communication channels) that are difficult and require additional care to maintain in the presence of asynchronous mobile corruptions.

Note that we leverage the advanced tool of full-information ACS to obtain a simplified constructive proof for showing the tightness of the feasible regimes of APSS. It remains an interesting open problem to further explore whether APSS with similar tight resilience can be achieved using more efficient ACS instantiations in alternative settings.

B Analysis of Resilience-Optimal APSS Protocols

In the following sections, we separately establish the security of our APSS schemes under the partially malicious and malicious adversarial settings, respectively, to prove our tight feasibility theorems stated in Section 5. We first collect a lemma that holds under both adversary models and subsequently invoke it in both security analyses. We then conclude this appendix with an analysis of the round and communication complexities of the constructions.

[17] In our constructions, each ACS instance is executed among the parties in $\mathcal{P}_{\mathsf{act}}$ and must tolerate up to f corrupted parties. Hence, the resilience condition requires $f \leq \lfloor \frac{|\mathcal{P}_{\mathsf{act}}|-1}{3} \rfloor$, For the partially malicious construction, where $|\mathcal{P}_{\mathsf{act}}| = t+f+1$, this requires $t \geq 2f$; for the malicious construction, where $|\mathcal{P}_{\mathsf{act}}| = t + 2f + 1$, it requires $t \geq f$.These conditions are implicit whenever the ACS properties are invoked below.

B.1 Common Lemmas

Lemma 9. *Assuming* ACS *is instantiated in the full-information setting [35], during the execution of the* Reshare *protocol, if* $|\mathcal{P}_{\mathsf{act}}| \leq n$*, all honest parties in* $\mathcal{P}_{\mathsf{act}}$ *that output from* $\mathsf{ACS}_{\mathsf{epk}}$ *will record an identical key set* $\{\mathsf{epk}_j\}_{j\in[|\mathcal{P}_{\mathsf{act}}|]}$*. Moreover, for every honest party* $\mathcal{P}_j \in \mathcal{P}_{\mathsf{act}}$ *(even if it does not provide an input to* $\mathsf{ACS}_{\mathsf{epk}}$*) that processes* T_{epk} *holds a secret key* esk_j *consistent with the recorded* epk_j*, i.e., for every message* m*,* $\mathsf{Dec}(\mathsf{esk}_j, \mathsf{Enc}(\mathsf{epk}_j, m)) = m$

Proof. By the *Agreement* property of ACS in the full-information setting, if $|\mathcal{P}_{\mathsf{act}}| \leq n$, all honest parties in $\mathcal{P}_{\mathsf{act}}$ output from $\mathsf{ACS}_{\mathsf{epk}}$ will output the same set T_{epk}. By the *external validity* of $\mathsf{ACS}_{\mathsf{epk}}$, every key included in T_{epk} is valid. According to the description, for any index $j \in [|\mathcal{P}_{\mathsf{act}}|]$, if $\mathsf{epk}'_j \in T_{\mathsf{epk}}$, all honest parties in $\mathcal{P}_{\mathsf{act}}$ will record it as the epk_j. Otherwise, honest parties in $\mathcal{P}_{\mathsf{act}}$ will take a default $\mathsf{epk}^{\mathsf{def}}$ as epk_j. Since both T_{epk} and the default key $\mathsf{epk}^{\mathsf{def}}$ are common to all parties, the resulting public-key vector $\mathsf{epk}_{j\,j\in[|\mathcal{P}_{\mathsf{act}}|]}$ is identical across all honest parties in $\mathcal{P}_{\mathsf{act}}$.

Furthermore, each party initializes its local key pair to $(\mathsf{esk}^{\mathsf{def}}, \mathsf{epk}^{\mathsf{def}})$. If an honest $\mathcal{P}_j \in \mathcal{P}_{\mathsf{act}}$ provides its private input, generates $(\mathsf{esk}'_j, \mathsf{epk}'_j)$, and its fresh key epk'_j is included in T_{epk}, then it sets $\mathsf{esk}_j \leftarrow \mathsf{esk}'_j$; otherwise it uses $\mathsf{esk}_j = \mathsf{esk}^{\mathsf{def}}$. In particular, a party that provides no private input proposes no fresh key and uses the default secret key. In either case, the secret key corresponding to the recorded epk_j is locally known to $\mathcal{P}_j$. □

B.2 Security Analysis for Partially Malicious Secure APSS

To prove our tight feasibility statements in the partially malicious setting, we will first establish the security of our APSS scheme against the partially malicious adversary under the SINGLE model. Specifically, we analyze correctness, liveness, and privacy in turn. Then, for each security property, after presenting the analysis under the SINGLE model, we further provide a brief sketch showing how it extends to the BOTH model.

Correctness

Lemma 10 (Correctness against partially malicious adversary under SINGLE model). *If* $t \leq n-f-1$*,* $\mathcal{A}^{\mathsf{part}}$*'s advantage in the correctness game* G_S^C *is negligible. Specifically, any honest party that runs the* reconstruction *protocol and outputs will output* s *that the dealer input in the* Share *algorithm, except with negligible probability.*

Proof. In the partially malicious setting, $|\mathcal{P}_{\mathsf{act}}| = t+f+1$. If $t \leq n-f-1$, then $|\mathcal{P}_{\mathsf{act}}| \leq n$ always holds. From Lemma 9, all honest parties in $\mathcal{P}_{\mathsf{act}}$ use a consistent public-key set and can correctly decrypt the encrypted subshares included in the output of $\mathsf{ACS}_{\mathsf{share}}$. Then we prove the correctness by induction on the epoch number τ:

(1) $\tau = 0$. After the initial Share phase, all honest parties in $\mathcal{P}_{\mathsf{act}}$ will get a valid share such that for any $t+1$-sized index set T_s:

$$\sum_{k\in T_s} \mu_k s_k^{(0)} = s.$$

(2) $\tau > 0$. Assume that in epoch τ every honest party in $\mathcal{P}_{\mathsf{act}}$ providing a private input to $\mathsf{Reshare}^{(\tau\to\tau+1)}$ holds a valid $s_k^{(\tau)}$ $s_k^{(\tau)}$ such that:

$$\sum_{k\in T^*} \mu_k s_k^{(\tau)} = s.$$

From the *Agreement* property of ACS, all honest parties in $\mathcal{P}_{\mathsf{act}}$ that output will receive the same T_s, containing $t+1$ sets of encrypted subshares $\{c_{j,k}^{(\tau)}\}_{k\in[|\mathcal{P}_{\mathsf{act}}|]}$ from distinct parties $\mathcal{P}_j$. An honest dealer submits a transcript only after receiving this private input, and a partially malicious adversary cannot modify its stored states, so every selected dealer reshares its valid epoch-τ share. Let $I_s \leftarrow \{j \mid \{c_{j,k}^{(\tau)}\} \in T_s\}$. By Lemma 9, all honest parties in $\mathcal{P}_{\mathsf{act}}$

can decrypt a subshare $s_{j,i}^{(\tau)}$ for all dealers $\mathcal{P}_j$ that $j \in I_s$, such that for any $t+1$ honest parties (denoted by R):

$$\sum_{j \in R} \lambda_j [s_k^{(\tau)}]_j = s_k^{(\tau)}.$$

Then according to the description of the Reshare protocol, all honest parties that output in $\mathcal{P}_{\mathsf{act}}$ can interpolate their new shares in epoch $\tau+1$. We have:

$$\sum_{k \in I_s} \mu_k [s_k^{(\tau)}]_j = s_j^{(\tau+1)}.$$

Therefore:

$$\sum_{j \in R} \lambda_j s_j^{(\tau+1)} = \sum_{j \in R} \lambda_j (\sum_{k \in I_s} \mu_k [s_k^{(\tau)}]_j) = \sum_{k \in I_s} \mu_k s_k^{(\tau)} = s.$$

Thus, if $t \leq n-f-1$, any honest party that runs the reconstruction protocol and outputs will output the original secret s that the dealer inputs in the Share algorithm, except with negligible probability. □

We now extend Lemma 10 to the Both model.

Lemma 11 (Correctness against partially malicious adversary under Both model). *If $t \leq n-f-1$, $\mathcal{A}^{\mathsf{part}}$'s advantage in the correctness game G_B^C is negligible.*

The proof follows essentially the same argument as in the Single model.

Liveness

Lemma 12. *In epoch $\tau = 0$, if $|\mathcal{P}_{\mathsf{act}}| \leq n$, at least $t+1$ parties provide valid inputs to both $\mathsf{ACS}_{\mathsf{epk}}$ and $\mathsf{ACS}_{\mathsf{share}}$ of the $\mathsf{Reshare}^{(0\to1)}$ protocol.*

Proof. During the Share protocol, the honest dealer directly distributes to each $\mathcal{P}_i$ its initial share. If $|\mathcal{P}_{\mathsf{act}}| \leq n$, every $\mathcal{P}_i \in \mathcal{P}_{\mathsf{act}}$ holds the initial share that contains a valid share of the secret s (together with its auxiliary information under malicious settings). Therefore, all honest $\mathcal{P}_i \in \mathcal{P}_{\mathsf{act}}$ hold valid inputs at the beginning of epoch 0 and can submit well-formed inputs to both $\mathsf{ACS}_{\mathsf{epk}}$ and $\mathsf{ACS}_{\mathsf{share}}$ in $\mathsf{Reshare}^{(0\to1)}$. Since the number of honest parties in epoch 0 is at least $|\mathcal{P}_{\mathsf{act}}| - f \geq t+1$, at least $t+1$ parties provide valid inputs. □

Lemma 13. *If $t \leq n-f-1$, in every epoch τ, at least $t+1$ parties can provide inputs to each $\mathsf{ACS}_{\mathsf{epk}}$ and $\mathsf{ACS}_{\mathsf{share}}$ of the Reshare protocol.*

Proof. In the partially malicious setting, $|\mathcal{P}_{\mathsf{act}}| = t+f+1$. If $t \leq n-f-1$, then $|\mathcal{P}_{\mathsf{act}}| \leq n$ always holds. We discuss this Lemma in two cases:
(1) $\tau = 0$, this directly follows from Lemma 12.
(2) $\tau > 0$. By the induction hypothesis, at least $t+1$ parties provide valid inputs to both ACS instances of $\mathsf{Reshare}^{(\tau-1\to\tau)}$, so both instances terminate. By ACS Agreement and Termination, every honest participant eventually obtains the same T_{epk} and T_s when the honest messages are delivered. A party whose local epoch is $\tau' \leq \tau-1$ need not provide a private input to that instance: the ACS instances are initialized independently of its private input, and Lemma 9 supplies the default decryption key if it proposes no fresh key. It therefore decrypts its subshares, computes its refreshed share, and advances directly to epoch τ.

In the partially malicious setting, the adversary is not allowed to tamper with or delete the internal state and incoming messages of corrupted parties. Recall that our ACS is instantiated following the BKR framework using Bracha reliable broadcast [13] and full-information asynchronous consensus [35]. At the protocol level, all relevant communication in the underlying construction is performed via broadcasts. Moreover, every local progress condition can be satisfied after receiving messages from at most $|\mathcal{P}_{\mathsf{act}}| - f = t+1$ participating parties, without requiring the party's own outgoing messages to be delivered. Therefore, consider a party $\mathcal{P}_i$ corrupted under the partially malicious model, whose outgoing messages may

be omitted while its incoming messages and local state remain intact. Since at least $t+1$ honest parties participate normally, every protocol message broadcast by these parties and used to make progress is eventually received by P_i. Hence, P_i can satisfy the same progress conditions and continue executing the underlying ACS protocol even though none of its own outgoing messages are delivered. By *Termination* and *Agreement* of the underlying ACS construction, $\mathcal{P}_i$ eventually derives the same ACS output as the honest parties. Moreover, for any party in $\mathcal{P}_{\mathsf{act}}$ and corrupted in epoch $\tau-1$, it either stores a decryption key in its internal state for decrypting subshares in epoch $\tau-1$, or it is specified to use a default $\mathsf{esk}^{\mathsf{def}}$. Both cases allow it to decrypt sub-shares, compute shares in epoch τ, and advance to epoch τ. By Lemma 10, every such party holds a valid share of the secret s. Hence, all $|\mathcal{P}_{\mathsf{act}}| = t+f+1$ parties possess valid shares at the beginning of epoch τ. During epoch τ, the adversary may further corrupt up to f of these parties and prevent them from providing valid inputs. Therefore, at least $(t+f+1)-f = t+1$ parties still provide valid inputs to the ACS instances in epoch τ. □

Lemma 14 (Liveness against partially malicious adversary under SINGLE model). *If $t \leq n-f-1$, $\mathcal{A}^{\mathsf{part}}$'s advantage in the liveness game G_S^L is negligible. Specifically, any honest party that runs the* reconstruction *protocol can output a value, except with negligible probability.*

Proof. From Lemma 13, all parties in $\mathcal{P}_{\mathsf{act}}$ can output refreshed shares in every epoch τ. Under the SINGLE model, at most f additional parties may be corrupted before the shares are refreshed again. Hence, when reconstruction is invoked, at least $|\mathcal{P}_{\mathsf{act}}|-f = t+1$ honest parties can multicast their correct shares. Therefore, any honest party that runs the reconstruction protocol can interpolate a t-degree polynomial and compute a secret value, except with negligible probability. □

We now extend Lemma 14 to the BOTH model.

Lemma 15 (Liveness against partially malicious adversary under BOTH model). *If $t \leq n-f-1$, $\mathcal{A}^{\mathsf{part}}$'s advantage in the liveness game G_B^L is negligible. Specifically, any honest party that runs the* reconstruction *protocol can output a value, except with negligible probability.*

Proof (sketch). The proof follows essentially the same argument as in the SINGLE model. If $t \leq n-f-1$, all parties in $\mathcal{P}_{\mathsf{act}}$ can output refreshed shares in every epoch τ. Under the Both model, if the adversary does not corrupt any party in $\mathcal{P}_{\mathsf{act}}$ in the current Reshare phase, at most f additional parties may be corrupted before the shares are refreshed again. When reconstruction is invoked, at least $|\mathcal{P}_{\mathsf{act}}|-f = t+1$ parties can multicast their correct shares and reconstruct a secret value. Therefore, the advantage of $\mathcal{A}^{\mathsf{part}}$ in the liveness game G_B^L is negligible. □

Privacy

Lemma 16 (Privacy against partially malicious adversary under SINGLE model). *If $t \geq 3f$, $\mathcal{A}^{\mathsf{part}}$'s advantage in the privacy game G_S^P is negligible, i.e.,* $\mathsf{Adv}^{G_S^P}(\mathcal{A}^{\mathsf{part}}) = |\Pr[\text{“}\mathcal{A}^{\mathsf{part}} \text{ wins”}] - \frac{1}{2}| = negl(\lambda)$.

Proof (sketch). We prove the privacy of the protocol by defining a sequence of games and showing that the adversary's advantage in distinguishing between any two consecutive games is negligible. Finally, we show that in the last game, the adversary's view is independent of the challenge bit b.

Let $\mathbb{C}^{(\tau)}$ denote the set of parties in $\mathcal{P}_{\mathsf{act}}$ corrupted in epoch τ, with $|\mathbb{C}^{(\tau)}| \leq f$. Let $\mathbb{L}^{(\tau)}$ denote the set of parties in $\mathcal{P}_{\mathsf{act}}$ that failed to update their keys during the Reshare protocol, with $|\mathbb{L}^{(\tau)}| = |\mathcal{P}_{\mathsf{act}}| - t - 1 = f$.

- **Game 0:** This is the original privacy game G_S^P. The challenger $\mathcal{C}$ picks $b \leftarrow_{\$} \{0,1\}$ and runs the protocol using the secret s_b.

– **Game i=1 to T :** Then for each $i = 1$ to T, the transition from Game $i-1$ to Game i focuses on the Reshare protocol for epoch $\tau = T - i$ (starting from the last epoch backwards). For game i, it is identical to Game $i-1$, except that:
  - For parties in $\{\mathbb{L}^{(\tau-1)} \cup \mathbb{C}^{(\tau-1)} \cup \mathbb{C}^{(\tau)}\}$, fix their input share s_k^τ to consist of the values in honest execution in the previous game. For parties in $\mathcal{P}_{\mathsf{act}}$ but not in $\{\mathbb{L}^{(\tau-1)} \cup \mathbb{C}^{(\tau-1)} \cup \mathbb{C}^{(\tau)}\}$, set their input shares as random values.
  - For parties in $\mathcal{P}_{\mathsf{act}}$ but not in $\{\mathbb{L}^{(\tau-1)} \cup \mathbb{C}^{(\tau-1)} \cup \mathbb{C}^{(\tau)}\}$, fix the subshares sent to $\mathcal{P}_j \in \{\mathbb{L}^{(\tau)} \cup \mathbb{C}^{(\tau)}\}$ to match their received values from the honest execution in the previous game, then send encryptions of 0 to all parties in $\mathcal{P}_{\mathsf{act}}$ but not in $\{\mathbb{L}^{(\tau)} \cup \mathbb{C}^{(\tau)}\}$.
  - The output shares for $\mathcal{P}_i \in \{\mathbb{C}^{(\tau)} \cup \mathbb{C}^{(\tau+1)} \cup \mathbb{L}^{(\tau)}\}$ remain fixed as in the previous game.

  Indistinguishability. For parties in $\{\mathbb{L}^{(\tau-1)} \cup \mathbb{C}^{(\tau-1)} \cup \mathbb{C}^{(\tau)}\}$, their shares are identical to the previous game. These are exactly the parties whose input shares are known to the adversary. The input shares that the adversary can learn in epoch τ is at most:

$$|\mathbb{L}^{(\tau-1)}| + |\mathbb{C}^{(\tau-1)}| + |\mathbb{C}^{(\tau)}| \leq 3f$$

  Since $t \geq 3f$, for other honest parties, their input shares remain hidden from the adversary prior to the reconstruction phase. Hence, the joint vector of input shares directly observed by the adversary is identical in the two games. The input share outside $\{\mathbb{L}^{(\tau-1)} \cup \mathbb{C}^{(\tau-1)} \cup \mathbb{C}^{(\tau)}\}$ do not directly enter the adversary's view.
  It remains to consider the subshares generated from these modified input shares. For every party in $\{\mathbb{L}^{(\tau)} \cup \mathbb{C}^{(\tau)}\}$, the corresponding plaintext subshares are fixed to be identical to those in the previous game. For any party in $\mathcal{P}_{\mathsf{act}}$ but not in $\{\mathbb{L}^{(\tau)} \cup \mathbb{C}^{(\tau)}\}$, the adversary never obtains the corresponding plaintext sub-shares or decryption key before these values are erased. Therefore, replacing encryptions of real sub-shares with encryptions of 0 is indistinguishable under IND-CPA security. Moreover, since the hybrid games are defined in reverse epoch order, the epoch-$(\tau+1)$ output shares of parties in $\mathbb{C}^{(\tau+1)}$ have already been fixed and remain identical to those in the previous game. Therefore, the adversary cannot distinguish Game i from Game $i-1$.
– **Game T+1:** Identical to Game T, except that in the Share protocol, the dealer distributes the epoch-0 input shares fixed in Game T to all parties.
  Indistinguishability. The adversary observes exactly the same epoch-0 shares as in Game T. Therefore, the adversary cannot distinguish Game T from Game $T+1$.

In Game $T+1$, all values outside the adversary's view are sampled independently of the challenge bit b. The only epoch-0 input shares directly observed by the adversary are those of parties in $\mathbb{C}^{(0)}$, where $|\mathbb{C}^{(0)}| \leq f \leq t$. By the perfect privacy of degree-t Shamir secret sharing, this entire vector of exposed shares is jointly uniform and statistically independent of b. All subsequent values visible to the adversary are either fixed by the preceding games or generated independently of b. Therefore, the adversary's complete view in Game $T+1$ is statistically independent of b.

Hence, for any partially malicious adversary $\mathcal{A}^{\mathsf{part}}$:

$$\mathsf{Adv}^{\text{Game } T+1}(\mathcal{A}^{\mathsf{part}}) = \left|\Pr[\text{“}\mathcal{A}^{\mathsf{part}} \text{ wins in Game } T+1\text{”}] - \frac{1}{2}\right| = 0$$

Moreover, for every Game $i \in \{0, \ldots, T+1\}$, the difference in $\mathcal{A}$'s winning probabilities between any two consecutive games is negligible. It follows that:

$$\mathsf{Adv}^{G_S^P}(\mathcal{A}^{\mathsf{part}}) = \left|\Pr[\text{“}\mathcal{A}^{\mathsf{part}} \text{ wins in Game 0”}] - \frac{1}{2}\right| \leq \sum_{i=0}^{T+1} \text{negl}(\lambda) = \text{negl}(\lambda).$$

□

We now extend Lemma 16 to the Both model.

Lemma 17 (Privacy against partially malicious adversary under BOTH model). *If $t \geq 2f$, $\mathcal{A}^{\mathsf{part}}$'s advantage in the privacy game G_B^P is negligible, i.e.,* $\mathsf{Adv}^{G_B^P}(\mathcal{A}^{\mathsf{part}}) = |\Pr[\text{“}\mathcal{A}^{\mathsf{part}} \text{ wins”}] - \frac{1}{2}| = negl(\lambda)$.

Proof (sketch). We prove privacy via a game-sequence argument analogous to that under the SINGLE model, and focus only on the differences in corruption counting. Under the BOTH model, corruptions occurring during the $\mathsf{Reshare}$ phase are charged to the corruption budgets of both adjacent epochs. Hence, the adversary's effective exposure across an epoch transition is more restricted, which strictly limits the number of distinct shares it can obtain in one epoch. More concretely, the total number of epoch-τ shares that the adversary can learn from corrupted parties is bounded by:

$$|\mathbb{C}_{\mathsf{reshare}}^{(\tau-1)} \cup \mathbb{C}^{(\tau)}| \leq f.$$

Then we have:

$$|\mathbb{L}^{(\tau-1)} \cup \mathbb{C}_{\mathsf{reshare}}^{(\tau-1)} \cup \mathbb{C}^{(\tau)}| \leq 2f.$$

Therefore, as long as $t \geq 2f$, these shares reveal no information about the underlying secret. The remaining analysis follows identically to Lemma 16. □

Based on the above conclusions, we can thus state the partially malicious security theorems for our APSS scheme (Theorem 5 and Theorem 6), which can be proven as follows:

Proof (Theorem 5). Correctness, liveness, and privacy follow directly from Lemmas 10, 14 and 16. Therefore, if $3f \leq t \leq n-f-1$, our partially-malicious-secure APSS scheme satisfies the security properties of APSS against a partially malicious adversary under the SINGLE model. □

Proof (Theorem 6). Correctness, liveness and privacy follow directly from Lemmas 11, 15 and 17. Therefore, if $2f \leq t \leq n-f-1$, our partially-malicious-secure APSS scheme satisfies the security properties of APSS against a partially malicious adversary under the BOTH model. □

B.3 Security Analysis for Malicious Secure APSS

To prove our tight feasibility statements in the fully malicious setting, we first analyze correctness, liveness, and privacy of our APSS scheme against the malicious adversary under the SINGLE model. Then, for each property, we provide a brief sketch explaining how the analysis naturally extends to the BOTH model.

Correctness

Lemma 18. *For every epoch τ, let $\mathsf{cm}^{(\tau)}$ denote the commitment vector output by Share when $\tau = 0$, or the common commitment vector output by $\mathsf{Reshare}^{(\tau-1\to\tau)}$ when $\tau > 0$. Then, during $\mathsf{Reshare}^{(\tau\to\tau+1)}$:*

1. *every honest party that fixes cm^* fixes $\mathsf{cm}^* = \mathsf{cm}^{(\tau)}$; in particular, all such honest parties fix the same cm^*; and*
2. *every honest party that outputs from the protocol obtains the same updated commitment vector cm'.*

Proof. We prove the lemma by induction on the epoch number τ.
(1) $\tau = 0$. All honest parties receive the same commitment vector $\mathsf{cm}^{(0)}$ from Share. Only parties in $\mathbb{C}^{(0)}$ can send incorrect epoch-0 commitment reports. Since $|\mathbb{C}^{(0)}| \leq f$, any $2f+1$ identical reports from distinct parties include a correct report. Hence every honest party that fixes cm^* fixes $\mathsf{cm}^* = \mathsf{cm}^{(0)}$.
(2) $\tau > 0$. Assume that all honest parties that output from $\mathsf{Reshare}^{(\tau-1\to\tau)}$ obtain the same commitment vector and adopt it as their epoch-τ public input, denoted by cm. Thus incorrect epoch-τ reports can originate only from parties in $\mathbb{C}^{(\tau-1)} \cup \mathbb{C}^{(\tau)}$, whose number

is at most $2f$. Any $2f+1$ identical reports from distinct parties therefore include a report containing cm. Consequently, every honest party that fixes cm^* fixes $\mathsf{cm}^* = \mathsf{cm}$.

In either case, all honest parties that proceed to validate proposals fix the same cm^*. Together with the common public-key vector established in Lemma 9, this ensures that they use the same external predicate for $\mathsf{ACS}_{\mathsf{share}}$. By the *Agreement* property of ACS, all honest parties that output from this instance obtain the same set T_s, containing $t+1$ transcripts.

By the *External-Validity* property of ACS, every transcript in T_s can pass the $\mathsf{VerifyPVSS}$ in the predicate function for $\mathsf{ACS}_{\mathsf{share}}$ and therefore contains Pedersen commitments to the subshares of $s_j^{(\tau)}$ and $\hat{s}_j^{(\tau)}$ according to the *Completeness* and *Soundness* of Verifiable PKE. Moreover, within each transcript, all Pedersen commitments correspond to points on a common polynomial of degree at most t. According to the description of $\mathsf{Reshare}$, all parties in $\mathcal{P}_{\mathsf{act}}$ use the same deterministic interpolation algorithm on the same set of commitment values drawn from T_s to compute the new commitments for epoch $\tau+1$. Since the input commitments and the interpolation procedure are identical at all honest parties, they compute the same resulting commitment vector cm'. □

Lemma 19. *If $t \le n-2f-1$, for every honest party $\mathcal{P}_i \in \mathcal{P}_{\mathsf{act}}$ who outputs from the* $\mathsf{Reshare}$ *protocol in epoch τ, the refreshed shares $s_i^{(\tau+1)}$ and $\hat{s}_i^{(\tau+1)}$ are consistent with the commitment $\mathsf{cm}'[i]$.*

Proof. In the malicious setting, $|\mathcal{P}_{\mathsf{act}}| = t+2f+1$. If $t \le n-2f-1$, then $|\mathcal{P}_{\mathsf{act}}| \le n$ always holds. By the *External-Validity* property of ACS, every proposal $\mathsf{transcript}_j$ contained in the output set T_s passes the $\mathsf{VerifyPVSS}$ algorithm of the underlying PVSS scheme. In particular, from the *Completeness* and *Soundness* of verifiable PKE, for each transcript $\mathsf{transcript}_j \in T_s$ and each party index $k \in [|\mathcal{P}_{\mathsf{act}}|]$, the ciphertext for $\mathcal{P}_k$ is accompanied by a valid NIZK proof π certifying that it encrypts the same share value that is committed in the corresponding Pedersen commitment.

By Lemma 18, all honest parties who get output from $\mathsf{ACS}_{\mathsf{share}}$ will obtain the same public commitment vector cm'. By Lemma 9, in all transcripts generated by honest parties, the public key used for encrypting the sub-shares will have corresponding correct private keys stored locally at the respective honest parties. Therefore, each honest party $\mathcal{P}_i$ who outputs from $\mathsf{ACS}_{\mathsf{share}}$ in the resharing phase can successfully invoke $\mathsf{Deliver}(\mathsf{transcript}_j, \mathsf{esk}_i)$ for every $\mathsf{transcript}_j \in T_s$ to extract the plaintext subshares $[s_j^{(\tau)}]_i$ and $[\hat{s}_j^{(\tau)}]_i$ that are consistent with the cm'.

Each honest party in $\mathcal{P}_{\mathsf{act}}$ extracts its own subshare from the transcript set T_s, and then applies Lagrange interpolation to reconstruct its shares. Let $I_s \leftarrow \{j \mid \mathsf{transcript}_j \in T_s\}$, we have:

$$s_i^{(\tau+1)} = \sum_{j \in I_s} \mu_j [s_j^{(\tau)}]_i, \qquad \hat{s}_i^{(\tau+1)} = \sum_{j \in I_s} \mu_j [\hat{s}_j^{(\tau)}]_i,$$

where $\{\mu_j\}_{j \in I_s}$ are the Lagrange coefficients at zero.

For each $\mathsf{transcript}_j \in T_s$, let:

$$v_{j,i} = g^{[s_j^{(\tau)}]_i} h^{[\hat{s}_j^{(\tau)}]_i}$$

denote the Pedersen commitment to $\mathcal{P}_i$'s subshare contributed by $\mathcal{P}_j$. The public commitment $\mathsf{cm}'[i]$ is computed by interpolation in the exponent:

$$\mathsf{cm}'[i] = \prod_{j \in I_s} v_{j,i}^{\mu_j} = g^{\sum_{j \in I_s} \mu_j [s_j^{(\tau)}]_i} \cdot h^{\sum_{j \in I_s} \mu_j [\hat{s}_j^{(\tau)}]_i} = g^{s_i^{(\tau+1)}} h^{\hat{s}_i^{(\tau+1)}}.$$

Hence, the refreshed shares $(s_i^{(\tau+1)}, \hat{s}_i^{(\tau+1)})$ are consistent with the public commitment $\mathsf{cm}'[i]$. □

Lemma 20. *In the* $\mathsf{reconstruction}$ *protocol for epoch τ, every honest party that fixes a commitment vector fixes the same commitment vector cm^*. In particular, when $\tau = 0$, this vector is the one generated by* Share*; when $\tau > 0$, $\mathsf{cm}^* = \mathsf{cm}'$, where cm' is the commitment vector output by* $\mathsf{Reshare}^{(\tau-1 \to \tau)}$.

Proof. During reconstruction, each party multicasts its local epoch-τ commitment vector and fixes cm^* only after receiving the same vector from $2f+1$ distinct parties.

When $\tau = 0$, all honest parties receive the same initial commitment vector from Share. Only parties in $\mathbb{C}^{(0)}$ can send incorrect epoch-0 commitment reports. Since $|\mathbb{C}^{(0)}| \leq f < 2f+1$, any $2f+1$ identical reports from distinct parties contain a correct report. Hence every honest party that fixes a commitment vector fixes the one generated by Share. When $\tau > 0$, by Lemma 18, every honest party in $\mathcal{P}_{\mathsf{act}}$ that outputs from $\mathsf{Reshare}^{(\tau-1\to\tau)}$ obtains the same commitment vector cm' and advances to epoch τ. Parties in $\mathbb{C}^{(\tau-1)}$ may have their local epoch-τ commitment state modified before this output is fixed, and parties in $\mathbb{C}^{(\tau)}$ may send arbitrary commitment reports during reconstruction. Thus the number of incorrect epoch-τ commitment reports can only be at most $|\mathbb{C}^{(\tau-1)} \cup \mathbb{C}^{(\tau)}| \leq 2f$.

Therefore, among any $2f+1$ identical epoch-τ commitment reports, at least one report is sent by a party whose epoch-τ commitment state is honest and equal to cm'. The fixed vector must therefore be cm'. Hence every honest party that fixes a commitment vector in reconstruction fixes the same cm^*, and for $\tau > 0$ this vector equals the commitment vector output by $\mathsf{Reshare}^{(\tau-1\to\tau)}$. □

Lemma 21 (Correctness against malicious adversary under SINGLE model). *If $t \leq n-2f-1$, the adversary's advantage in the correctness game G_S^C is negligible. Specifically, any honest party that runs the* reconstruction *protocol and outputs will output s that the dealer input in the* Share *algorithm, except with negligible probability.*

Proof. We prove the lemma by induction.
(1) $\tau = 0$. In epoch 0, the shares are generated through the Share algorithm. By the algorithm description, for every honest party $\mathcal{P}_j \in \mathcal{P}_{\mathsf{act}}$, the received shares $(s_j^{(0)}, \hat{s}_j^{(0)})$ are consistent with the commitment $\mathsf{cm}^{(0)}[j]$. Moreover, as $f(\cdot)$ and $\hat{f}(\cdot)$ are degree-t polynomials with constant term s and $\hat{s}$ respectively, the shares held by honest parties form a valid (n,t)-threshold sharing of the secret s and $\hat{s}$.
(2) $\tau > 0$. Assume that after $\mathsf{Reshare}^{(\tau-1\to\tau)}$, the parties in $\mathcal{P}_{\mathsf{act}} \setminus \mathbb{C}^{(\tau-1)}$ that output hold shares $(s_i^{(\tau)}, \hat{s}_i^{(\tau)})$ lying on the same degree-t sharing of the original secret s and blinding secret $\hat{s}$, and that $\mathsf{cm}^{(\tau)} = \{g^{s_i^{(\tau)}} h^{\hat{s}_i^{(\tau)}}\}_{i\in[|\mathcal{P}_{\mathsf{act}}|]}$is the corresponding public output commitment vector. Then consider the execution of $\mathsf{Reshare}^{(\tau\to\tau+1)}$. The parties in $\mathcal{P}_{\mathsf{act}} \setminus (\mathbb{C}^{(\tau-1)} \cup \mathbb{C}^{(\tau)})$ hold correct epoch-τ inputs. All honest parties that proceed to validate proposals first fix $\mathsf{cm}^* = \mathsf{cm}^{(\tau)}$ by Lemma 18. From the *External-Validity* of ACS, for each transcript $\mathsf{transcript}_j \in T_s$, the commitments lie on a degree-t polynomial in the exponent, and interpolating these commitments at zero yields a value $v_{j,0} = \mathsf{cm}^{(\tau)}[j] = g^{s_j^{(\tau)}} h^{\hat{s}_j^{(\tau)}}$. By the induction hypothesis, all such commitments correspond to shares of the same secret s. Interpolating any $t+1$ commitments(denoted by R) $\{v_{j,0}\}_{j\in R}$ at zero yields

$$\prod_{j\in R} \left(g^{s_j^{(\tau)}} h^{\hat{s}_j^{(\tau)}}\right)^{\lambda_j} = g^s h^{\hat{s}}$$

Then, according to the description of the Reshare, all honest parties choose the same $t+1$-sized T_s and interpolate the new commitment vector. Utilizing the homomorphic property of Pedersen commitment, we have:

$$\prod_{j\in I_s} (g^{[s_j^{(\tau)}]_i} h^{[\hat{s}_j^{(\tau)}]_i})^{\mu_j} = g^{s_i^{(\tau+1)}} h^{\hat{s}_i^{(\tau+1)}} = \mathsf{cm}'[i]$$

Therefore:

$$\prod_{i\in R} \left(\mathsf{cm}'[i]\right)^{\lambda_i} = \prod_{i\in R} \Big(\prod_{j\in I_s} (g^{[s_j^{(\tau)}]_i} h^{[\hat{s}_j^{(\tau)}]_i})^{\mu_j} \Big)^{\lambda_i} = \prod_{j\in I_s} \left(g^{s_j^{(\tau)}} h^{\hat{s}_j^{(\tau)}}\right)^{\mu_j} = g^s h^{\hat{s}}$$

Thus, the Reshare preserves the committed secret across epochs.

By Lemma 19, if $t \leq n-2f-1$, for each honest party $\mathcal{P}_i \in \mathcal{P}_{\mathsf{act}}$ that output from $\mathsf{Reshare}^{(\tau\to\tau+1)}$, the refreshed shares $s_i^{(\tau+1)}$ and $\hat{s}_i^{(\tau+1)}$ are consistent with the public commitment $\mathsf{cm}'[i]$. Thus, the underlying secret shared across epoch $\tau+1$ is identical to that of epoch τ.

By induction, for every epoch τ, all honest parties hold shares that are consistent with a degree-t sharing of the original secret s.

During reconstruction in epoch T, by Lemma 20, every honest party that fixes a commitment vector fixes the same vector $\mathsf{cm}^* = \mathsf{cm}'$ when $\tau > 0$, or the one generated by Share when $\tau = 0$. An honest party accepts a revealed pair $(s_j^{(T)}, \hat{s}_j^{(T)})$ only if

$$g^{s_j^{(T)}} h^{\hat{s}_j^{(T)}} = \mathsf{cm}^*[j].$$

Since $\mathsf{cm}^* = \mathsf{cm}'$ is the commitment vector corresponding to the sharing of the original secret s, the computational binding of Pedersen commitments implies that every accepted share equals the share committed at the corresponding index, except with negligible probability. Therefore, any $t+1$ accepted shares reconstruct the original secret s.

Consequently, in the correctness game G_S^C, the adversary can cause the honest parties to reconstruct a value different from s only with negligible probability. □

We now extend Lemma 21 to the Both model.

Lemma 22 (Correctness against malicious adversary under Both model). *If $t \leq n-2f-1$, the adversary's advantage in the correctness game G_B^C is negligible. Specifically, any honest party that runs the* reconstruction *protocol and outputs will output s that the dealer input in the* Share *algorithm, except with negligible probability.*

The proof follows essentially the same argument as in the Single model.

Liveness

Lemma 23. *In $\mathsf{Reshare}^{(\tau\to\tau+1)}$, every PVSS transcript submitted by an honest party to $\mathsf{ACS}_{\mathsf{share}}$ satisfies the external predicate at every honest party that evaluates the proposal, regardless of whether the evaluating party provides an input to this handoff.*

Proof. Let $\mathcal{P}_j$ be an honest party that submits a $\mathsf{transcript}_j$ to $\mathsf{ACS}_{\mathsf{share}}$. By the protocol description, before submitting $\mathsf{transcript}_j$, $\mathcal{P}_j$ fixes a commitment vector cm^*, processes a public-key vector $\{\mathsf{epk}_k\}_{k\in[|\mathcal{P}_{\mathsf{act}}|]}$, and verifies that $g^{s_j^{(\tau)}} h^{\hat{s}_j^{(\tau)}} = \mathsf{cm}^*[j]$. Since $\mathcal{P}_j$ follows the protocol, it generates $\mathsf{transcript}_j$ with respect to this public-key vector. The commitments in $\mathsf{transcript}_j$ correspond to polynomials of degree at most t in the exponent, and the constant commitment satisfies $v_{j,0} = g^{s_j^{(\tau)}} h^{\hat{s}_j^{(\tau)}} = \mathsf{cm}^*[j]$.

By Lemma 18, all honest parties that fix cm^* during this handoff fix the same commitment vector.by Lemma 9, all honest parties that output from $\mathsf{ACS}_{\mathsf{epk}}$ obtain the same public-key vector. Therefore, every honest party in $\mathcal{P}_{\mathsf{act}}$ that evaluates its proposal use the same cm^* and the same $\{\mathsf{epk}_k\}_{k\in[|\mathcal{P}_{\mathsf{act}}|]}$ as the inputs to the external predicate. By the completeness of the underlying verifiable PKE and NIZK systems, the ciphertexts and their accompanying proofs pass verification; the transcript satisfies the external predicate at every honest party that evaluates it. □

Lemma 24. *If $2f \leq t \leq n-2f-1$, then in every epoch τ, every honest party in $\mathcal{P}_{\mathsf{act}}$ that participates in $\mathsf{Reshare}^{(\tau\to\tau+1)}$ can eventually compute refreshed shares $(s_i^{(\tau+1)}, \hat{s}_i^{(\tau+1)})$, regardless of whether the evaluating party provides an input to this handoff.*

Proof. In the malicious setting, $|\mathcal{P}_{\mathsf{act}}| = t+2f+1$, if $t \leq n-2f-1$, then $|\mathcal{P}_{\mathsf{act}}| \leq n$ always holds. Then we will discuss the Lemma in two cases:
(1) $\tau = 0$. Since the dealer of Share is honest, every honest party in $\mathcal{P}_{\mathsf{act}}$ receives a correct initial private state and the same initial public commitment vector $\mathsf{cm}^{(0)}$. Since $|\mathcal{P}_{\mathsf{act}}| - f = (t+2f+1) - f = t+f+1 \geq 2f+1$, at least $2f+1$ honest parties can multicast the same commitment vector $\mathsf{cm}^{(0)}$ in the Reshare protocol. Therefore, every honest party can fix $\mathsf{cm}^* = \mathsf{cm}^{(0)}$.

All honest parties in $\mathcal{P}_{\mathsf{act}}$ that hold valid initial private inputs submit fresh public keys to $\mathsf{ACS}_{\mathsf{epk}}$. By *Termination* of ACS, all honest parties obtain T_{epk}. After obtaining T_{epk} and cm^*, these honest parties generate PVSS transcripts from their initial shares and submit

them to $\mathsf{ACS}_{\mathsf{share}}$. By Lemma 23, these transcripts satisfy the external predicate at every honest party that evaluates them. Hence at least $t+1$ parties provide valid inputs to each of $\mathsf{ACS}_{\mathsf{epk}}$ and $\mathsf{ACS}_{\mathsf{share}}$.

From *Termination* and *Agreement* of ACS, every honest party participating in $\mathsf{Reshare}^{(0\to1)}$ eventually obtains the same output set T_s. By Lemma 9, it holds the secret key corresponding to the public key recorded at its index, using the default key if it provides no fresh input. Therefore it can invoke $\mathsf{Deliver}(\mathsf{transcript}_j, \mathsf{esk}_i)$ for each selected transcript and interpolate its refreshed shares $(s_i^{(1)}, \hat{s}_i^{(1)})$.

(2) $\tau > 0$. Assume the statement holds for $\mathsf{Reshare}^{(\tau-1\to\tau)}$. Then all honest parties in $\mathcal{P}_{\mathsf{act}}$ outside $\mathbb{C}^{(\tau-1)}$ output $(s_i^{(\tau)}, \hat{s}_i^{(\tau)})$ from $\mathsf{ACS}_{\mathsf{share}}$ in epoch $\tau-1$. By Lemma 19, these shares are consistent with the output commitment vector, and by Lemma 18, this output commitment vector is the same for all honest parties; denote it by $\mathsf{cm}^{(\tau)}$.

Parties corrupted in epoch $\tau-1$ may have had their epoch-τ states modified, and parties corrupted in epoch τ may fail to act honestly in the current handoff. Thus, excluding parties in $\mathbb{C}^{(\tau-1)} \cup \mathbb{C}^{(\tau)}$, at least $|\mathcal{P}_{\mathsf{act}}| - 2f = (t+2f+1) - 2f = t+1$ honest parties still hold epoch-τ shares consistent with $\mathsf{cm}^{(\tau)}$. All these $t+1$ honest parties will broadcast the same commitment vector $\mathsf{cm}^{(\tau)}$. Since $t+1 \geq 2f+1$, this is enough to include $2f+1$ parties with the same commitment vector, and every honest party can fix $\mathsf{cm}^* = \mathsf{cm}^{(\tau)}$. Then, by the same argument as in the base case, at least $t+1$ honest parties submit PVSS transcripts to $\mathsf{ACS}_{\mathsf{share}}$, and every honest party participating in $\mathsf{Reshare}^{(\tau\to\tau+1)}$ interpolates its refreshed shares $(s_i^{(\tau+1)}, \hat{s}_i^{(\tau+1)})$. □

Lemma 25 (Liveness against malicious adversary under SINGLE model). *$\mathcal{A}^{\mathsf{Byz}}$'s advantage in the liveness game G_S^L is negligible. Specifically, if $2f \leq t \leq n-2f-1$, any honest party in $\mathcal{P}_{\mathsf{act}}$ that runs the* reconstruction *protocol can output a value, except with negligible probability.*

Proof. When $\tau = 0$, since the dealer of Share is honest, every honest party in $\mathcal{P}_{\mathsf{act}}$ receives a correct initial private state and the same initial commitment vector. Therefore, at least $|\mathcal{P}_{\mathsf{act}}| - f = t+f+1 \geq t+1$ honest parties in $\mathcal{P}_{\mathsf{act}}$ hold valid epoch-0 shares.

When $\tau > 0$, by Lemma 24, every honest party in $\mathcal{P}_{\mathsf{act}}$ that participates in $\mathsf{Reshare}^{(\tau-1\to\tau)}$ eventually obtains refreshed epoch-τ shares and the updated public auxiliary information, regardless of whether it held a valid internal state. Consequently, all parties in $\mathcal{P}_{\mathsf{act}} \setminus \big(\mathbb{C}^{(\tau-1)} \cup \mathbb{C}^{(\tau)}\big)$ hold valid epoch-τ shares and act honestly during reconstruction. Their number is at least $|\mathcal{P}_{\mathsf{act}}| - \big|\mathbb{C}^{(\tau-1)} \cup \mathbb{C}^{(\tau)}\big| \geq |\mathcal{P}_{\mathsf{act}}| - 2f = t+1$.

Thus, in either case, at least $t+1$ honest parties in $\mathcal{P}_{\mathsf{act}}$ hold valid epoch-τ shares and the corresponding commitment vector. These parties multicast the same commitment vector during reconstruction. Since $t \geq 2f$, upon eventual delivery of the honest messages, every honest party participating in reconstruction, whether or not it belongs to $\mathcal{P}_{\mathsf{act}}$, receives at least $2f+1$ identical commitment reports and fixes a commitment vector. By Lemma 20, all such parties fix the same correct cm^*.

The same $t+1$ honest parties reveal their epoch-τ shares. By Lemmas 19 and 20, each revealed pair $(s_j^{(\tau)}, \hat{s}_j^{(\tau)})$ satisfies $g^{s_j^{(\tau)}} h^{\hat{s}_j^{(\tau)}} = \mathsf{cm}^*[j]$ and is therefore accepted by every honest party participating in reconstruction. So every honest party can eventually receive at least $t+1$ valid shares and can interpolate an output value. □

We now extend Lemma 25 to the BOTH model.

Lemma 26 (Liveness against malicious adversary under BOTH model). *$\mathcal{A}^{\mathsf{Byz}}$'s advantage in the liveness game G_B^L is negligible. Specifically, if $2f \leq t \leq n-2f-1$, any honest party in $\mathcal{P}_{\mathsf{act}}$ that runs the* reconstruction *protocol can output a value, except with negligible probability.*

Proof (sketch). The proof follows essentially the same argument as in the SINGLE model. Under the Both model, the adversary may have deleted the epoch-τ incoming messages of at most f parties, including messages used in $\mathsf{Reshare}^{(\tau\to\tau+1)}$, even if these parties are released before the handoff begins. These parties are therefore not required to obtain epoch-$(\tau+1)$

shares. All $|\mathcal{P}_{\mathsf{act}}| - f = t + f + 1$ remaining parties in $\mathcal{P}_{\mathsf{act}}$ eventually obtain the handoff outputs and compute valid refreshed shares.

Since corruptions during $\mathsf{Reshare}^{(\tau \to \tau+1)}$ are charged to the corruption quotas of both adjacent epochs, at least $(|\mathcal{P}_{\mathsf{act}}| - f) - f = t + 1$ honest parties hold valid epoch-$(\tau+1)$ shares. As $t \geq 2f$, these parties also provide at least $2f + 1$ identical correct commitment reports. So every honest party fixes the same correct cm^* and eventually receives $t+1$ valid revealed shares, which suffice to interpolate an output value. □

Privacy

Lemma 27 (Privacy against malicious adversary under SINGLE model). *If $t \geq 4f$, $\mathcal{A}^{\mathsf{Byz}}$'s advantage in the privacy game G_S^P is negligible, i.e., $\mathsf{Adv}^{G_S^P}(\mathcal{A}^{\mathsf{Byz}}) = |\Pr[\text{“}\mathcal{A}^{\mathsf{Byz}} \text{ wins”}] - \frac{1}{2}| = negl(\lambda)$.*

Proof (sketch). We prove the privacy of the protocol by defining a sequence of games and showing that the adversary's advantage in distinguishing between any two consecutive games is negligible. Finally, we show that in the last game, the adversary's view is independent of the challenge bit b.

Let $\mathbb{C}^{(\tau)}$ denote the set of parties in $\mathcal{P}_{\mathsf{act}}$ corrupted in epoch τ, with $|\mathbb{C}^{(\tau)}| \leq f$, and let $\mathbb{L}^{(\tau)}$ denote the set of parties in $\mathcal{P}_{\mathsf{act}}$ that failed to update their keys during the $\mathsf{Reshare}$ protocol, with $|\mathbb{L}^{(\tau)}| = |\mathcal{P}_{\mathsf{act}}| - t - 1 = 2f$.

We work in the common reference string (CRS) model where the CRS contains generators g, h such that the Pedersen commitment trapdoor $a = \log_g h \in \mathbb{F}^*$ is known only to the challenger. This allows equivocation of Pedersen commitments while preserving public consistency. Additionally, since the NIZK system is zero-knowledge, the challenger can simulate proofs without access to witnesses.

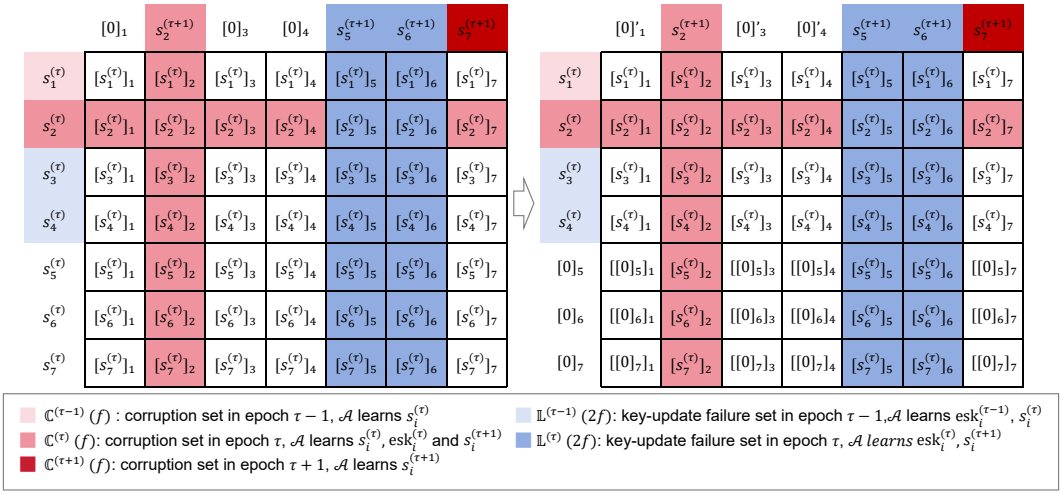


Fig. 8. Example illustrating the transition from Game $i.1$ to Game $i.2$ under the boundary setting $n = 6f + 1$ and $t = 4f$ (with $f = 1$). It depicts one resharing ($\tau \to \tau + 1$) and highlights the adversary's view during the game transition. Parties are grouped and color-coded according to their corruption status as summarized in the legend. For each group, the plaintext shares or subshares observable to the adversary are marked in the figure. Although the challenger modifies the underlying shares of honest parties in Game $i.2$, the set of plaintext information visible to the adversary remains unchanged compared to Game $i.1$.

– **Game 0:** This is the original privacy game G_S^P. The challenger $\mathcal{C}$ picks $b \leftarrow_{\$} \{0, 1\}$ and runs the protocol using the secret s_b.

– Then for each $i = 1$ to T, the transition from Game $i-1$ to Game i focuses on the resharing for epoch $\tau = T - i$ (starting from the last epoch backwards). We define sub-games as follows:

- **Game** $i.1$**:** Identical to Game $i-1$, except that for the epoch $\tau = T - i$, the NIZK proofs in the PVSS transcripts of honest parties are replaced with simulated proofs generated by the zero-knowledge simulator.
  Indistinguishability. By the *zero-knowledge* property of the Verifiable PKE scheme, simulated proofs are computationally indistinguishable from real proofs. Therefore, the adversary cannot distinguish Game $i.1$ from Game $i-1$.
- **Game** $i.2$ **(Game** i**):** Identical to Game $i.1$, except that for the Reshare protocol in the target epoch $\tau = T - i$:
  * For parties in $\{\mathbb{C}^{(\tau)} \cup \mathbb{C}^{(\tau-1)} \cup \mathbb{L}^{(\tau-1)}\}$, fix their input share $(s_k^{(\tau)}, \hat{s}_k^{(\tau)})$ to the corresponding values in the previous game. Let $\phi^{(\tau)}(x)$ and $\hat{\phi}^{(\tau)}(x)$ denote the degree-t sharing polynomials defining the epoch-τ input shares in the previous game. Interpolates a t-degree polynomial $\phi'^{(\tau)}(x)$ that passes through the points $\phi'^{(\tau)}(k) = \phi^{(\tau)}(k)$ for $\mathcal{P}_k \in \{\mathbb{C}^{(\tau)} \cup \mathbb{C}^{(\tau-1)} \cup \mathbb{L}^{(\tau-1)}\}$ and $\phi'^{(\tau)}(0) = 0$. It then defines $\hat{\phi}'^{(\tau)}(x) = \hat{\phi}^{(\tau)}(x) + a^{-1}\big(\phi^{(\tau)}(x) - \phi'^{(\tau)}(x)\big)$ For all honest parties $\mathcal{P}_k \in \mathcal{P}_{\mathsf{act}}$, sets their modified epoch-τ input shares for the epoch to $s_k'^{(\tau)} = \phi'^{(\tau)}(k), \hat{s}_k'^{(\tau)} = \hat{\phi}'^{(\tau)}(k)$.
  * For each honest party $\mathcal{P}_k \in \mathcal{P}_{\mathsf{act}}$ but not in $\{\mathbb{C}^{(\tau)} \cup \mathbb{C}^{(\tau-1)} \cup \mathbb{L}^{(\tau-1)}\}$, let $\phi_k^{(\tau)}(x)$ and $\hat{\phi}_k^{(\tau)}(x)$ denote its resharing polynomials in the previous game. The challenger interpolates a degree-t polynomial $\phi_k'^{(\tau)}(x)$ satisfying $\phi_k'^{(\tau)}(0) = s_k'^{(\tau)}$ and $\phi_k'^{(\tau)}(j) = \phi_k^{(\tau)}(j)$ for every $\mathcal{P}_j \in \{\mathbb{C}^{(\tau)} \cup \mathbb{L}^{(\tau)}\}$ It then defines $\hat{\phi}_k'^{(\tau)}(X) = \hat{\phi}_k^{(\tau)}(X) + a^{-1}\big(\phi_k^{(\tau)}(X) - \phi_k'^{(\tau)}(X)\big)$. The modified subshares are defined as $[s_k'^{(\tau)}]_j = \phi_k'^{(\tau)}(j), [\hat{s}_k'^{(\tau)}]_j = \hat{\phi}_k'^{(\tau)}(j)$.
  * The output shares $(s_j^{(\tau+1)}, \hat{s}_j^{(\tau+1)})$ for $\mathcal{P}_i \in \{\mathbb{C}^{(\tau)} \cup \mathbb{C}^{(\tau+1)} \cup \mathbb{L}^{(\tau)}\}$ remain fixed as in the previous game.
  * The challenger uses the trapdoor to ensure all public sub-commitments and the new commitment list are identical to the honest execution.

  Indistinguishability. We analyze the transition from Game $i.1$ to Game $i.2$ by considering, in turn, the adversary's view of the input shares, the subshares exchanged during the Reshare protocol, and the remaining public transcript.

  Input shares. First, consider the Reshare input shares in epoch τ. For all parties in $\{\mathbb{C}^{(\tau)} \cup \mathbb{C}^{(\tau-1)} \cup \mathbb{L}^{(\tau-1)}\}$, their input shares pair $(s_k^{(\tau)}, \hat{s}_k^{(\tau)})$ are fixed to be identical to those in Game $i.1$. These are exactly the parties whose input shares are known to the adversary either in the current epoch, in the previous epoch, or due to key-update failure.
  Since the number of the input shares that the adversary can learn in epoch τ is at most:
  $$|\mathbb{C}^{(\tau)} \cup \mathbb{C}^{(\tau-1)} \cup \mathbb{L}^{(\tau-1)}| \leq f + f + 2f = 4f$$
  the challenger can sample the modified degree-t sharing polynomial subject to the prescribed constant term and these fixed evaluations.
  Hence, the joint vector of input shares exposed to the adversary is identical in Game $i.1$ and Game $i.2$.

  Subshares during the Reshare protocol. Next, consider the subshares sent during the Reshare protocol. For each honest party in $\mathcal{P}_{\mathsf{act}}$, the subshares sent to parties in $\{\mathbb{C}^{(\tau)} \cup \mathbb{L}^{(\tau)}\}$ are fixed to match the values received in Game $i.1$. These are exactly the subshares observed by the adversary in epoch τ. The remaining subshares are defined by interpolating a t-degree polynomial consistent with the fixed subshares and the new input share. Therefore, the distribution of all subshares visible to the adversary is identical in the two games.

  Commitments and other encrypted subshares. Finally, we consider the adversary's public view, which consists of all public commitments and the encrypted

subshares sent to parties whose decryption keys are not known to the adversary during the Reshare protocol.
Using the trapdoor of the Pedersen commitment scheme, since $h = g^a$, the definitions of the modified blinding polynomials give

$$g^{\phi'^{(\tau)}(X)} h^{\hat{\phi}'^{(\tau)}(X)} = g^{\phi^{(\tau)}(X)} h^{\hat{\phi}^{(\tau)}(X)}$$

and, for every modified honest dealer P_k,

$$g^{\phi_k'^{(\tau)}(X)} h^{\hat{\phi}_k'^{(\tau)}(X)} = g^{\phi_k^{(\tau)}(X)} h^{\hat{\phi}_k^{(\tau)}(X)}.$$

Consequently, the input commitments, all sub-commitments, and the resulting new commitment vector are identical to those in Game $i.1$. The NIZK proofs remain simulated as in Game $i.1$.
For the encrypted subshares sent by honest parties during the Reshare protocol, although the underlying plaintext subshares of honest parties are modified in Game $i.2$, all such subshares are transmitted in encrypted form under the public keys of the recipients. Since the adversary does not possess the corresponding decryption keys for parties not in $\{\mathbb{C}^{(\tau)} \cup \mathbb{L}^{(\tau)}\}$, its view consists only of ciphertexts encrypting the honest parties' shares. By the IND-CPA security of the encryption scheme, encryptions of the modified shares in Game $i.2$ are computationally indistinguishable from encryptions of the original shares in Game $i.1$.
Combining the above arguments, the adversary cannot distinguish Game $i.2$ from Game $i.1$.

- **Game T+1:** Identical to Game T, except that in the Share protocol, the dealer takes 0 as the secret and send $(s_k'^{(0)}, \hat{s}_k'^{(0)})$ generated in the previous game for all parties.
  Indistinguishability. By the construction of the previous hybrid games, after processing epoch $\tau = 0$, the input shares of all parties already form a valid sharing of the value 0. In Game $T+1$, the challenger explicitly modifies the Share to share the value 0 with exactly the same shares as in Game T. Since the resulting shares, commitments, and messages are identically distributed in the two games, the adversary's view in Game $T+1$ is identical to that in Game T. Therefore, the adversary cannot distinguish Game T from Game $T+1$.

Since in Game $T+1$ the secret shared in the Share protocol is fixed to 0, and all subsequent messages and internal states are generated independently of the challenge bit b, the adversary's entire view in Game $T+1$ is independent of b. Hence, for any PPT adversary $\mathcal{A}$:

$$\mathsf{Adv}^{\text{Game } T+1}(\mathcal{A}) = \left| \Pr[\text{“}\mathcal{A}^{\mathsf{Byz}} \text{ wins in Game } T+1\text{”}] - \frac{1}{2} \right| = 0$$

Moreover, for every Game $i \in \{0, \ldots, T+1\}$, the difference in $\mathcal{A}$'s winning probabilities between any two consecutive games is negligible. It follows that:

$$\mathsf{Adv}^{G_S^P}(\mathcal{A}^{\mathsf{Byz}}) = \left| \Pr[\text{“}\mathcal{A}^{\mathsf{Byz}} \text{ wins in Game 0”}] - \frac{1}{2} \right| \leq \sum_{i=0}^{T+1} \mathrm{negl}(\lambda) = \mathrm{negl}(\lambda)$$

□

Lemma 28 (Privacy against malicious adversary under Both model). *If $t \geq 3f$, $\mathcal{A}^{\mathsf{Byz}}$'s advantage in the privacy game G_B^P is negligible, i.e., $\mathsf{Adv}^{G_B^P}(\mathcal{A}^{\mathsf{Byz}}) = |\Pr[\text{“}\mathcal{A}^{\mathsf{Byz}} \text{ wins”}] - \frac{1}{2}| = negl(\lambda)$.*

Proof (sketch). We prove privacy via a game-sequence argument analogous to that under the Single model, and focus only on the differences in corruption counting. Under the Both model, corruptions occurring during the Reshare phase are charged to the corruption budgets of both adjacent epochs. Hence, the adversary's effective exposure across an epoch

transition is more restricted, which strictly limits the number of distinct shares it can obtain in one epoch. More concretely, the total number of epoch-τ shares that the adversary can learn from corrupted parties is bounded by:

$$|\mathbb{C}_{\mathsf{reshare}}^{(\tau-1)} \cup \mathbb{C}^{(\tau)}| \leq f.$$

Then we have:

$$|\mathbb{L}^{(\tau-1)} \cup \mathbb{C}_{\mathsf{reshare}}^{(\tau-1)} \cup \mathbb{C}^{(\tau)}| \leq 3f.$$

Therefore, as long as $t \geq 3f$, these shares reveal no information about the underlying secret. The remaining analysis follows identically to Lemma 27.

Having established correctness, liveness, and privacy, we are ready to conclude the security of our APSS scheme against a malicious adversary, which can be proven as follows.

Proof (Theorem 7). Correctness, liveness, and privacy follow directly from Lemmas 21, 25 and 27, respectively. Therefore, if $4f \leq t \leq n - 2f - 1$, our malicious-secure APSS scheme satisfies the security properties of APSS against a malicious adversary under the SINGLE model. □

Proof (Theorem 8). Correctness, liveness, and privacy follow directly from Lemmas 22 , 26 and 28, respectively. Therefore, if $3f \leq t \leq n - 2f - 1$, our malicious-secure APSS scheme satisfies the security properties of APSS against a malicious adversary under the BOTH model. □

B.4 Complexity Analysis for APSS

We briefly summarize the round and communication complexities of our two APSS constructions under the full-information ACS instantiation. Each invocation of Reshare sequentially invokes two ACS instances. At the optimal-resilience points in the SINGLE model, namely $n = 4f + 1, t = 3f$ for the partially malicious construction and $n = 6f + 1, t = 4f$ for the malicious construction, we use the underlying full-information agreement with a corruption ratio bounded away from the extremal 1/3 point. Using the construction of Huang et al. [35], both constructions therefore achieve expected $\widetilde{O}(n^4)$ asynchronous rounds and $\widetilde{O}(n^8) + O(\lambda n^4)$ communication cost per invocation of Reshare, where λ denotes the security parameter.

The same bounds apply to the malicious construction in the BOTH model, whose optimal-resilience point is $n = 5f + 1, t = 3f$. For the partially malicious construction in the Both model, the boundary point is $n = 3f + 1, t = 2f$, where the underlying full-information agreement operates at the extremal resilience threshold. Under the same instantiation, the resulting complexity becomes expected $\widetilde{O}(n^{12})$ asynchronous rounds and $\widetilde{O}(n^{16}) + O(\lambda n^4)$ communication cost per invocation of Reshare.

Given the high complexities of our resilience-optimal APSS constructions, an important open problem is to develop more efficient constructions achieving optimal resilience without resorting to heavyweight full-information ABA protocols.