New Results in Adaptively-Secure Broadcast Encryption and Succinct Computational Secret Sharing from Standard Lattices

Rishab Goyal
UW-Madison*

Saikumar Yadugiri
UW-Madison†

Abstract

Projective pseudorandom generators (PPRGs), introduced by Applebaum et al. [ABI+23], expand a short seed into a long pseudorandom string, with a special feature that one can generate a short *projected* key, for any set S, such that it reveals the output bits in S while preserving pseudorandomness of the remaining bits. They demonstrated that PPRGs can bypass the 'representation size barrier' in computational secret sharing schemes, and designed PPRGs assuming hardness of a variety of standard cryptographic assumptions. Later, Hsieh, Waters, and Wu [HWW25] showed that PPRGs can also be used to design adaptively-secure broadcast encryption in the standard model, without relying on unprovable heuristics.

In this work, we design succinct computational secret sharing schemes for graph access structures and adaptively-secure flexible broadcast encryption in the standard model, both with optimal parameters. The main driver for both our results is a new construction of a compact programmable pseudorandom set sampler (PPS), introduced by Goyal-Yadugiri [GY26c], from standard Learning with Errors (LWE) assumption [Reg05]. We show that PPS generically imply PPRGs with many useful properties and, moreover, by replacing PPRGs with PPS we can achieve full succinctness in the underlying applications to secret sharing and adaptive security compilers for broadcast encryption. This yields the *first* flexible broadcast encryption scheme in the standard model from *any* falsifiable assumption with fully compact ciphertext size, and a succinct computational secret sharing for evolving graph access structures with optimal share size from standard LWE.

*Email: rishab@cs.wisc.edu
†Email: saikumar@cs.wisc.edu

Contents

1 Introduction **3**

2 Technical Overview **7**
- 2.1 Programmable Pseudorandom Set Samplers . . . 7
- 2.2 Prior Constructions and the Compression Problem . . . 8
- 2.3 Our PPS from LWE . . . 9
- 2.4 Application I: Flexible Broadcast Encryption in the Standard Model . . . 11
- 2.5 Application II: Projective PRGs and Secret Sharing for Graphs . . . 12

3 Preliminaries **15**
- 3.1 Lattice Preliminaries . . . 16

4 Programmable Pseudorandom Set Sampler **17**
- 4.1 Definition . . . 17
- 4.2 Preliminaries . . . 18
- 4.3 Construction . . . 20
- 4.4 Full Compactness Analysis . . . 23
- 4.5 Mode Indistinguishability Analysis . . . 24
- 4.6 Payload Hiding Analysis . . . 30
- 4.7 A More Efficient Construction for Polynomially Bounded Users . . . 31

5 Application I: Broadcast Encryption **32**
- 5.1 Security Analysis . . . 34
- 5.2 Distributed BE and Alternatives . . . 42

6 Application II: Projective PRGs and Succinct Computational Secret Sharing **43**
- 6.1 Publicly Sampleable Projective PRGs from PPS . . . 43
- 6.2 Succinct CSS for Graph Access Structures from Robust PPS . . . 49
  - 6.2.1 CSS for Evolving Graphs . . . 51
- 6.3 Constructing Robust PPS . . . 55

A Q-Bounded Robust PPS from PPS **66**
- A.1 Bounded Adaptive Robustness Analysis . . . 68
- A.2 Secret Sharing for Evolving Graphs from PPS . . . 71

1 Introduction

Projective pseudorandom generators (PPRGs), introduced by Applebaum, Beimel, Ishai, Kushilevitz, Liu, and Vaikuntanathan [ABI+23], expand a *short* seed sd into a *long* pseudorandom string y where, for any set S of output positions, there is a short *projected key* sd_S that reveals exactly the bits in S while every other bits stay pseudorandom. Simply put, sd_S reveals only $\{y_i : i \in S\}$, while hiding $\{y_i : i \notin S\}$.

PPRGs are a new primitive that overcome the established *representation size barrier* [BL90, KW93, Csi95, Csi96, LVW18, LV18, ABN+22, ABI+23] in secret sharing schemes at the cost of relaxing to computationally bounded adversaries. Essentially, PPRGs are a useful abstraction to design succinct computational secret sharing (SCSS). [ABI+23] designed PPRGs with different levels of succinctness under a variety of computational assumptions (RSA/DDH/bilinear-DDH/LWE/obfuscation), and used them to construct SCSS for graph access structures, monotone CNF formulas, and (less succinctly) subclasses of monotone circuits. Hsieh, Waters, and Wu [HWW25] observed that a mild generalization of PPRGs, where projected keys can be sampled publicly, suffices for designing adaptively-secure broadcast encryption (BE) [FN94] via the beautiful two-key compiler of Gentry-Waters [GW09] without relying on random oracle heuristic [BR93]. Concretely, [HWW25] obtained the first standard-model compiler to boost semi-static security to (full) adaptive security for BE and its decentralized strengthenings [WQZD10, BZ14].

PPRGs & the *full* succinctness problem. While prior works [ABI+23, HWW25, BDH+26, HJW26] established PPRGs as a valuable building block across seemingly unrelated contexts in cryptography such as succinct secret sharing and adaptive security compilers, the known lattice instantiations leave compactness limitations in these applications (even in an asymptotic sense). Broadly, this limitation can be traced to one of the two following reasons: (a) either the length of projected keys is super-polylogarithmic in the set size $|S|$, or (b) even fully-succinct projected keys can not preserve the succinctness via existing compilers. We elaborate on this below.

***Inflexibility* in broadcast encryption.** Flexible broadcast encryption (FBE) [FWW23, GLWW23] is a popular variant of classic BE that supports fully decentralized key generation. Each user samples its own key pair, without any synchronization or knowledge of others, and a sender encrypts to any list of public keys with a ciphertext whose size is independent of the length of the list. While recent works [GY26a, AMR26] designed the first FBE scheme from a falsifiable cryptographic assumption, both of these constructions crucially rely on the random oracle (RO) heuristic [BR93]. This is because we only have a singular approach to prove (full) adaptive security in BE, which is the popular Gentry-Waters two-key compiler [GW09, FWW23] that requires ROs. While a standard model version for Gentry-Waters compiler exists [HWW25], it builds on PPRGs where the public parameters of compiler BE system need to scale with the output length of the PPRG. Thus, even with fully-optimal PPRG (whose size of projected key grows poly-logarithmically with set size), its application in designing adaptively-secure BE incurs at least a linear cost in the number of users, N. Thus, while being useful to replace ROs, PPRGs do not preserve succinctness in the context of broadcast encryption.

Sub-optimal SCSS via *post-quantum* PPRGs and more. A secret sharing scheme for graph access structures is defined such that users are associated with vertices of the graph, and any two users who share an edge are authorized to learn the secret. In other words, any coalition

of users that do not share an edge must learn nothing about the secret. Such structures capture natural policies. In dual control, an operation requires two authorized employees and the allowed pairs form a graph determined by roles and conflicts of interest. Applebaum et al. [ABI+23] showed that PPRGs directly give a secret sharing scheme for such access structures, where the (per-user) share size is exactly the length of a single projected key. However, the only known constructions for PPRGs that give fixed (i.e., polylogarithmic) size projected keys relied either on the RSA assumption or indistinguishability obfuscation. Unfortunately, these constructions do not give compact graph shares from standard falsifiable lattice assumptions. Francati and Venturi [FV25] extend PPRG-based secret sharing to evolving graphs using RSA or iO and SSB hashing. Our goal is to obtain optimal share size from standard LWE.

In summary, PPRGs, while useful for many new applications, do not have optimal post-quantum constructions (i.e., need either RSA or obfuscation), and their usage in various applications brings in unwanted restrictions (e.g., bounded restriction in BE and static declaration of access structure for SCSS). We address all these limitations in this work and design the first fully-secure flexible broadcast encryption as well as fully-optimal SCSS for graph access structures under falsifiable lattice assumptions, thereby offering plausible post-quantum security.

Our Results

The core technical tool behind our results is the notion of *programmable pseudorandom set sampler* (PPS) [GY26c]. A PPS is a dual-mode functional encryption scheme [BSW11, GKW18], where a secret key is associated with a set S and a ciphertext is associated with an index i. The special property about PPS ciphertexts is that they can be generated in either a public-key mode or a secret-key mode, where public-key ciphertexts only encrypt junk messages, and the two modes are indistinguishable as long as secret-key ciphertexts encrypt random messages or are undecryptable. More precisely, the two encryption modes must be indistinguishable for adversaries who: (a) hold a secret key for S that satisfies the challenge index, but the encrypted message is random (called *mode indistinguishability*) and (b) hold a secret key for S that does *not* satisfy the challenge index, irrespective of the encrypted message (called *payload hiding*).

A crucial property that PPS must achieve is the compactness of secret keys, which means that the length of secret keys should be independent of the set S, as otherwise PPS are trivial to design from one-way functions. We adapt the PPS definition to an exponential index space and show that a PPS generically implies a PPRG with many interesting properties such as robustness against multiple projected keys, unbounded output space, and a fast (local) evaluation. We refer the reader to Sections 4 and 6 for further details.

Result 1 – PPS from LWE with full compactness. Existing PPS constructions [GY26c] support only a polynomially bounded index space, where at least one of the scheme parameters (either master key size or ciphertext length) grows at least linearly in N, which is the total number of supported indices. Unfortunately, PPS schemes with sub-optimal compactness are insufficient for any of the aforementioned applications. To enable our applications, we give a new PPS construction where every parameter is polylogarithmic in the N, and every algorithm runs in time polynomial in $\lambda, \log N$ under polynomial hardness of standard LWE assumption [Reg05].

Theorem 1.1 (Informal)**.** Assuming the hardness of LWE, there exists a PPS for the index space $[N]$ in which the common random string, the master secret key, every ciphertext, and every set key have size $\mathsf{poly}(\lambda, \log N)$. Moreover, public-key ciphertexts are uniformly random strings.

This construction requires careful combination of various entities from homomorphic evaluation and garbled circuit literature. In Section 2, we explain why this avoids the compression issue briefly discussed as the main technical barrier to compact PPS in [GY26c].

Result 2 – Adaptively secure FBE in the standard model. We generalize the two-key compiler of [GW09, FWW23] so that the selector bits come from a PPS rather than from a random oracle or a PPRG. This yields the following result.

Theorem 1.2 (Informal)**.** Any semi-statically secure FBE scheme can be compiled, using a PPS, into an adaptively secure FBE scheme in the standard model. The compiler at most doubles the public keys and ciphertexts of the underlying scheme and adds $\mathsf{poly}(\lambda)$ bits to the common random / reference string, each public key, and each ciphertext.

Both [GY26a] and [AMR26] obtain their adaptively secure FBE schemes in the random oracle model by first constructing a semi-statically secure FBE scheme from decomposed LWE with $\mathsf{poly}(\lambda)$ parameters and then applying the Gentry-Waters compiler. The semi-statically secure scheme of [GY26a] uses a random oracle whereas that of [AMR26] is in the standard model. Instantiating our compiler with the latter and the PPS of Theorem 1.1, we obtain the following.

Corollary 1.3 (Informal)**.** Assuming polynomial hardness of decomposed LWE assumption, there exists an adaptively secure FBE scheme in the standard model in which the common reference string, public keys, secret keys, and ciphertexts all have size $\mathsf{poly}(\lambda)$.

This is the first adaptively secure FBE scheme in the standard model from a falsifiable assumption. We compare the state-of-the art on FBE with our scheme in Table 1. Our compiler also applies to a slotted distributed broadcast encryption scheme (DBE) and adds only $\mathsf{poly}(\lambda)$ overhead to the crs. Moreover, our PPS plugs into the standard-model equivocal DBE of [GY26c] and removes the 1-key LWE assumption from it, giving adaptively secure DBE with transparent setup, keys and ciphertexts of size $\mathsf{poly}(\lambda)$, and a crs of size $N \cdot \mathsf{poly}(\lambda)$ from decomposed LWE (Corollary 5.22). This matches the standard-model DBE result of [AMR26] through matrix commitments [Wee25, WW25] rather than through the expensive tree encoding framework.

Result 3 – Projective PRGs and secret sharing for graphs. We show that a PPS gives a publicly sampleable PPRG [HWW25] and, following [ABI+23], we boost it to provide robustness against any polynomial number Q of projected keys by secret sharing every output across many PPS instances.

Theorem 1.4 (Informal)**.** Assuming PPS, there exists a publicly sampleable PPRG with output length ℓ that is robust against Q adaptively chosen projected keys, with projected keys of size $\mathsf{poly}(\lambda, \log \ell)$ and public parameters of size $Q \cdot \ell \cdot \mathsf{poly}(\lambda, \log \ell)$. Evaluating one output reads only $\mathsf{poly}(\lambda, \log \ell)$ bits of the public parameters.

The last property, which we call *locality*, is absent from the PPRGs from LWE, DDH, and bilinear maps of [ABI+23, HWW25], where evaluating one output reads the public values of every index in the projected set. Locality is useful for secret sharing as the dealer can hand every party only the part of the public parameters it needs and avoid public information altogether. Plugging the above PPRG into the graph secret sharing scheme of [ABI+23], every graph with L vertices and independence number at most Q gets a scheme with no public information and shares of size $Q \cdot \mathsf{poly}(\lambda, \log L)$. This beats the shares of size $L^{1/2} \cdot \mathsf{poly}(\lambda)$ of the plain LWE construction of [ABI+23] for a swathe of parameters, but it is still not optimal.

Instead of going through a PPRG, we directly use PPS directly for computational secret sharing for evolving graphs, following the evolving access structures of [KNY16] and the computational setting of Francati and Venturi [FV25]. Vertices join one at a time, a joining vertex declares its edges to the vertices already present and receives a share, and the existing shares are not modified. For the same swathe of parameters as above, a PPS gives secret sharing for evolving graphs with collusion bound Q (i.e., the number shares that are corrupted are at most Q), no public share, and shares of size $Q \cdot \mathsf{poly}(\lambda)$ independent of the number of vertices. However, this is still not optimal.

The reason is that using PPS, we can only argue security against one key corruption, which forces us to deal with bounded robustness, whereas SCSS requires full robustness. To deal with this, we define a generalization and relaxation of PPS called *robust* PPS. A robust PPS scheme is like a PPS scheme except that it does not need the mode indistinguishability property and the payload hiding property holds against multiple secret key corruptions. Using robust PPS, we get the following result.

Theorem 1.5 (Informal)**.** Assuming a robust PPS, there exists a computational secret sharing scheme for evolving graphs with unbounded collusion where every share has size $\mathsf{poly}(\lambda)$ independent of the number of vertices.

We then show two ways to construct robust PPS, one from batched IBE [AFP25] and one from ABE with succinct secret keys [SW05, GPSW06, BGG+14]. The former gives instantiations from pairings and the latter from LWE. [ABI+23] prove the security of their secret sharing schemes in a static model, where the graph and the corrupted set are fixed before the shares are generated, and Lu and Waters [LW25] show that their graph scheme is adaptively secure for a graph fixed in advance. We prove all our schemes in a stronger model, where the adversary chooses the arrivals, the edges, and the corrupted vertices adaptively [FGV25]. Table 2 summarizes our results on secret sharing for graph access structures.

Reference	Assumption	Adaptive	No RO
[FWW23]	Witness Encryption + LWE	✗	✓
[AMR26]	Decomposed LWE	✗	✓
[AMR26]	Decomposed LWE	✓	✗
[GY26a]	Decomposed LWE	✓	✗
This Work	Decomposed LWE	✓	✓

Table 1: Comparison with prior distributed and flexible broadcast encryption schemes with optimal parameter sizes and transparent crs. **No RO** denotes the absence of random oracles.

Related work. Distributed broadcast encryption [WQZD10, BZ14] lets every user generate its own key pair for a slot in a bounded set of users, and lattice-based DBE was studied in [CW24, CHW25, AMR25, WW25, GY26c, AMR26, CW26]. Flexible broadcast encryption was introduced by Freitag-Waters-Wu [FWW23] as a generalization of DBE with no bound on the number of users and no slots. [GLWW23] show how to compile a DBE scheme into a weaker version of FBE. The works of Goyal-Yadugiri [GY26a] and Abram-Malavolta-Roy [AMR26] give the first FBE schemes from falsifiable lattice assumptions, both adaptively secure in the random oracle model. [GY26c, GY26a, GY26b] develop the equivocal encryption framework for adaptive security.

Hsieh-Waters-Wu [HWW25] give a standard model compiler from semi-static to adaptive security for centralized BE and DBE with a bounded number of users. In their compiler, the master

Reference	Assumption	Share size	EV	PQ
[ABI+23]	RSA / $i\mathcal{O}$ + SSB	1	✗	✗
[FV25]	RSA	1	✓	✗
[ABI+23]	DDH	$n^{2/3}$	✗	✗
[ABI+23]	LWE	$n^{1/2}$	✗	✓
Theorem A.11†	PPS (LWE)	a	✓	✓
Corollary 6.24	q-type bilinear	$1^\star$	✓‡	✗
Corollary 6.29	LWE	1	✓	✓

Table 2: Computational secret sharing for the access structure of a graph on n vertices. Share sizes hide $\mathsf{poly}(\lambda, \log n)$ factors. a bounds the independence number of the graph. $\star$ denotes the presence of a public share. † denotes that the number of shares that can be revealed is bounded by a. **EV** denotes support for evolving graphs, where ‡ denotes a bounded requirement for vertex neighborhood size. **PQ** denotes a post-quantum instantiation.

public key or the crs of the compiled scheme grows linearly with the number of users. In the centralized setting, these public parameters can be moved out of the master public key into a common reference string, from which a public pre-processing step derives the projected key and the two recipient sets, so that the master public key stays short.

Projective PRGs were introduced by [ABI+23] as a tool for succinct computational secret sharing. Recently, Branco et al. [BDH+26] expanded their use cases and used PPRGs to construct hidden bits generators with batch openings and succinct common reference string. Hsieh, Jain, and Waters [HJW26] identified enhanced properties of PPRGs, instantiated them from RSA, and obtained batched threshold encryption and a secret-key BE with short ciphertexts. These constructions are not post-quantum.

2 Technical Overview

In this section, we provide a high-level overview of our results. We start by recalling programmable pseudorandom set samplers (PPS) and the role they play in our two applications. We then describe the two constructions of [GY26c], isolate the compression problem that neither of them resolves, and present our construction from LWE (Section 2.3). The last two subsections describe our applications to flexible broadcast encryption (Section 2.4) and to projective PRGs and secret sharing (Section 2.5). Throughout, we focus on conveying the main technical ideas rather than precise definitions or parameter choices.

2.1 Programmable Pseudorandom Set Samplers

A PPS scheme [GY26c] for a message space $\mathcal{M}$ and an index space $[N]$ consists of six algorithms. The global setup $\mathsf{GSetup}(1^\lambda, N)$ outputs a common random string crs, where N is given in binary, so the index space may be exponentially large, and MSKGen samples a master secret key msk. A ciphertext for an index $i \in [N]$ is produced either by $\mathsf{skEnc}(\mathsf{msk}, i, \mu)$, which encrypts a message $\mu \in \mathcal{M}$, or by $\mathsf{pkEnc}(i)$, which uses no secret at all. $\mathsf{KGen}(\mathsf{msk}, S)$ outputs a short key sk_S for a set $S \subseteq [N]$, and $\mathsf{Decode}(S, \mathsf{sk}_S, i, \mathsf{ct})$ recovers μ from a secret-key ciphertext for index i whenever $i \in S$. The scheme is *fully compact* if crs, msk, every ciphertext, and every key have size $\mathsf{poly}(\lambda, \log N)$. There are two security requirements for PPS which are expressed as games. In both security games, the attacker first receives crs and a challenge ciphertext for an index i^* of its choice, then asks for

secret-key ciphertexts $\mathsf{skEnc}(\mathsf{msk}, i, \mu_i)$ for any indices $i \neq i^*$ and messages μ_i of its choice, and only then chooses the set S for which it receives sk_S.

Mode indistinguishability. If $i^* \in S$, then $\mathsf{pkEnc}(i^*)$ is indistinguishable from $\mathsf{skEnc}(\mathsf{msk}, i^*, \mu)$ for a message μ *chosen at random by the challenger*.

Payload hiding. If $i^* \notin S$, then $\mathsf{pkEnc}(i^*)$ is indistinguishable from $\mathsf{skEnc}(\mathsf{msk}, i^*, \mu)$ for any message μ chosen by the attacker.

The restriction to a random μ in the first property is necessary, since an attacker holding sk_S with $i^* \in S$ can simply decode. The definition of PPS in [GY26c] fixes N in unary. We strengthen it to an exponential index space, which the flexible setting needs (Section 4).

PPS as a replacement for a programmable random oracle. In our applications, every per-index value in the real system is produced by pkEnc, which needs no secret and, in our construction, outputs a uniformly random string. In a security proof, the reduction *programs* the value at an index i by switching it to $\mathsf{skEnc}(\mathsf{msk}, i, \mu)$ for a message μ of its choice, and whether this is detectable depends only on whether the key that is eventually handed out covers i. If it does, mode indistinguishability applies and the reduction can only program a random message, which is all a Gentry-Waters style reduction needs for a user inside the challenge set. If it does not, payload hiding applies and the reduction can program any message, which is what it needs for a user outside the challenge set. These are exactly the two cases in which the two-key compiler of [GW09] programs its random oracle.

PPS versus projective PRGs. A projective PRG (PPRG) [ABI+23] expands a seed into a string $\mathbf{y} \in \{0,1\}^\ell$ such that a short projected key for a set S reveals exactly $(y_i)_{i \in S}$ while the other bits remain pseudorandom. A publicly sampleable PPRG (PSPRG) [HWW25] additionally allows a projected key to be sampled without the seed, indistinguishably from a real one. Any PPS gives a PSPRG. Let the public parameters consist of crs together with a secret-key ciphertext of a random bit y_i for every index i, let the seed be msk, let $\mathsf{Project}(\mathsf{msk}, S) := \mathsf{KGen}(\mathsf{msk}, S)$, and let $\mathsf{Eval}(S, \mathsf{sk}_S, i) := \mathsf{Decode}(S, \mathsf{sk}_S, i, \mathsf{ct}_i)$. Payload hiding says that the outputs outside S are pseudorandom given sk_S. A projected key for S can be publicly sampled as a key under a fresh master secret key, and mode indistinguishability together with payload hiding shows that this is indistinguishable from an honest projection (Section 6.1). The converse does not appear to hold. A PSPRG only hides the outputs *outside* the projected set, whereas payload hiding experiment of PPS lets a reduction choose the messages at every index other than the target and still obtain a key for a set containing those indices. This difference is what our applications exploit, and it is why the FBE compiler of Section 2.4 does not seem to go through with a PSPRG.

2.2 Prior Constructions and the Compression Problem

Goyal and Yadugiri [GY26c] gave two constructions of PPS, both for a polynomially bounded index space.

PPS from LWE with long ciphertexts. The master secret key consists of $2N$ LWE secrets $\mathbf{s}_{j,b}$ for $j \in [N]$ and $b \in \{0,1\}$, and the key for a set S with characteristic vector $(r_1, \ldots, r_N)$ is $\mathsf{sk}_S = \sum_j \mathbf{s}_{j,r_j}$. A ciphertext for index i contains $\mathbf{c}_0 = \mathbf{A}\mathbf{x}$ for a short $\mathbf{x}$ and one scalar $c_{j,b} \approx (\mathbf{s}_{j,b}^\top \mathbf{A})\mathbf{x} + z_j$ for each pair (j,b), where the masks z_j sum to zero and the entry $c_{i,1}$ additionally carries a rounding term that hides the message. Correctness is a telescoping sum and programmability is per index,

since every index has its own secrets. The price is that msk and every ciphertext have $\Theta(N)$ entries. For an exponential index space the scheme is not even well defined.

PPS from 1-key LWE with short ciphertexts. The natural way to compress the ciphertext is the key-homomorphic machinery of [BGG+14]. Let f_S be the set membership circuit of S over $\lceil \log N \rceil$ input bits and $\mathbf{B}_S = \mathsf{EvalF}(\mathbf{B}, f_S)$. The master secret key holds a matrix $\mathbf{A}$ with a trapdoor, a matrix $\mathbf{B}$, and a vector $\mathbf{d}$. The key for S is a short preimage $\mathbf{r}_S$ such that $[\mathbf{A} \mid \mathbf{B}_S]\mathbf{r}_S = \mathbf{d}$. A ciphertext for index i is $(\mathbf{s}^\top \mathbf{A} + \mathsf{err}, \mathbf{s}^\top(\mathbf{B} - \mathsf{encvec}(i) \otimes \mathbf{G}) + \mathsf{err}, \lceil \mathbf{s}^\top \mathbf{d} \rfloor_2 + \mu)$ where $\mathsf{encvec}(i) \in \{0,1\}^{\lceil \log N \rceil}$ is the bit vector encoding of $i \in [N]$. Everything has size $\mathsf{poly}(\lambda, \log N)$, and payload hiding follows from LWE as in the selective proof of [BGG+14]. Mode indistinguishability is the problem. The reduction receives the first two ciphertext components as an LWE challenge, so it holds no trapdoor for $\mathbf{A}$, yet it must answer an *adaptively* chosen set S such that $i^* \in S$ with a preimage $\mathbf{r}_S$ of $\mathbf{d}$. If S were declared selectively, the reduction could sample $\mathbf{r}_S$ first and set $\mathbf{d} := [\mathbf{A} \mid \mathbf{B}_S]\mathbf{r}_S$. For an adaptive S, no way to do this without a trapdoor is known. [GY26c] introduced 1-key LWE, a new falsifiable assumption that postulates exactly the required indistinguishability.

The compression problem. The two constructions sit on opposite sides of one obstacle. Per-index secrets make programming easy and ciphertexts long. Compressing the ciphertext ties all indices to a single digest $\mathbf{B}_S$, and the key then becomes a trapdoor-sampled preimage with respect to that digest, so the adaptive choice of S collides with the LWE challenge in the usual way. We need to compress the per-index ciphertext to $\mathsf{poly}(\lambda, \log N)$ bits without turning the key into a preimage that only a trapdoor can produce.

2.3 Our PPS from LWE

Our observation is that the dependence between the key and the ciphertext can be reversed. Rather than making the key a preimage of a digest, we let the key carry the digest, and we let the ciphertext carry a function of the digest. The crs is a matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times \lceil \log N \rceil m}$, the digest of a set S is $\mathbf{A}_S = \mathsf{EvalF}(\mathbf{A}, f_S)$ for the set membership circuit f_S, and the ciphertext for index i contains a program that, given any digest $\mathbf{A}_S$, outputs an LWE encoding of μ that can be decoded if and only if $f_S(i) = 1$. The program must hide the LWE secret, so we garble it.

The construction. We use a reusable garbling scheme Garb with a secret garbling key gk, in which a garbled input $\widetilde{x} = \mathsf{InpGarb}(\mathsf{gk}, x)$ can be evaluated on any number of garbled circuits $\widetilde{C} = \mathsf{CircGarb}(\mathsf{gk}, C)$, together with a somewhere equivocal PRF (SEPRF) [HJO+16] with key key. The master secret key is $\mathsf{msk} = (\mathsf{key}, \mathsf{gk})$ and contains no lattice trapdoor. The key for a set S is $\mathsf{sk}_S = (\mathsf{key}, \widetilde{\mathbf{A}}_S)$, where $\widetilde{\mathbf{A}}_S = \mathsf{InpGarb}(\mathsf{gk}, \mathbf{A}_S)$ is the garbled digest. To encrypt μ for index i, sample an LWE secret $\mathbf{s}$, a matrix $\mathbf{P}$, a wide smudging vector $\widetilde{\mathbf{e}}$, and a vector $\mathsf{tg} \in \mathbb{Z}_q^\lambda$ whose entries avoid the rounding boundaries, and consider the circuit C that hardwires these values and on input a digest $\mathbf{B}$ outputs $\mathbf{P}$, the string $h = \lceil \mathsf{tg} \rfloor_2 \oplus \mu$, and

$$\mathbf{c}^\top = \mathbf{s}^\top(\mathbf{A} - \mathsf{encvec}(i) \otimes \mathbf{G}) + \mathsf{err}, \qquad \mathbf{d}^\top = \mathbf{s}^\top \mathbf{B}\mathbf{G}^{-1}(\mathbf{P}) - \mathbf{s}^\top \mathbf{P} + \widetilde{\mathbf{e}}^\top + \mathsf{tg}.$$

where $\mathsf{encvec}(i) \in \{0,1\}^{\lceil \log N \rceil}$ is the bit vector encoding of $i \in [N]$. The $\mathsf{skEnc}(\mathsf{msk}, i, \mu)$ ciphertext is $\mathsf{ct} = (t, T)$ where $t \leftarrow \{0,1\}^\lambda$ and $T = \mathsf{SEPRF.Eval}(\mathsf{key}, t) \oplus \mathsf{CircGarb}(\mathsf{gk}, C)$. $\mathsf{pkEnc}(i)$ outputs a uniformly random (t, T). To decode with sk_S, unmask the garbled circuit using key and evaluate it on $\widetilde{\mathbf{A}}_S$ to obtain $(\mathbf{c}^\top, \mathbf{P}, \mathbf{d}^\top, h)$. Then compute $\mathbf{H}_{S,i} = \mathsf{EvalFX}(\mathbf{A}, f_S, \mathsf{encvec}(i))$ and $\mathsf{tg}' = \mathbf{d}^\top - \mathbf{c}^\top \mathbf{H}_{S,i} \mathbf{G}^{-1}(\mathbf{P})$, and output $h \oplus \lceil \mathsf{tg}' \rfloor_2$.

Correctness is exactly the key-homomorphic identity $(\mathbf{A} - \mathsf{encvec}(i) \otimes \mathbf{G})\mathbf{H}_{S,i} = \mathbf{A}_S - f_S(i)\mathbf{G}$ of [BGG+14]. If $i \in S$, then $\mathbf{c}^\top \mathbf{H}_{S,i}\mathbf{G}^{-1}(\mathbf{P}) \approx \mathbf{s}^\top \mathbf{A}_S \mathbf{G}^{-1}(\mathbf{P}) - \mathbf{s}^\top \mathbf{P}$, so $\mathsf{tg}' \approx \mathsf{tg}$ up to noise of norm $B \ll q$, and the rounding is unaffected because tg can be sampled in a way that tg avoids the boundaries. The circuit f_S has depth $O(\log N)$, so $\|\mathbf{H}_{S,i}\| \leq m^{O(\log N)}$. We need LWE with a sub-exponential modulus-to-noise ratio due to error smudging. The circuit C has input length $nm\lceil \log q \rceil$ and size $\mathsf{poly}(\lambda, n, m, \log N, \log q)$ independently of S, i, and μ, so the garbled circuit, the ciphertext, and the key all have size $\mathsf{poly}(\lambda, \log N)$, and no algorithm runs in time growing with N beyond reading the description of S.

Why these ingredients. Garbling hides $\mathbf{s}$ and the noise from the decoder, which learns only $C(\mathbf{A}_S)$, and reusability [WW24] lets one garbling key serve every ciphertext while each key garbles its digest once. If Garb was also a *blind* garbled circuit scheme [BLSV18, WW24], its blindness property says that a simulated garbled circuit whose output is uniformly random is itself a uniformly random string. Thus, once we have argued that $(\mathbf{c}^\top, \mathbf{P}, \mathbf{d}^\top, h)$ is pseudorandom, the entire ciphertext is pseudorandom, which is the distribution of pkEnc. This is also what makes the per-user values in our applications uniformly random strings. The SEPRF provides the necessary toolkit to argue adaptivity. In the mode indistinguishability game, the reduction hands out the challenge ciphertext (t^*, T^*) before it knows S, but the simulated garbled circuit depends on $\mathbf{A}_S$. The SEPRF lets the reduction output a uniformly random T^* at the start and, after S arrives, simulate $\widetilde{C}^*$ from $\mathbf{A}_S$ and derive a key key with $\mathsf{SEPRF.Eval}(\mathsf{key}, t^*) = T^* \oplus \widetilde{C}^*$ that agrees with the equivocation key on every other input. This equivocated key is what goes into sk_S. In other words, the ciphertext commits to nothing, and the key, which is generated after S is known, explains it. This is the equivocation principle of [GY26c, GY26a], applied at the level of the PRF with no random oracle and no lattice trapdoor. We now prove the two security properties of PPS below.

Mode indistinguishability. Let $i^* \in S$. Starting from a secret-key challenge ciphertext, we use the following hybrids to argue this property,

(1) Switch key to an equivocated key, which lets us sample T^* uniformly at setup and defer the sampling of C^* and key to the challenge phase by security of SEPRF,

(2) Replace $\widetilde{C}^*$ by a simulated garbling of the output $C^*(\mathbf{A}_S)$, by security of Garb,

(3) Use $f_S(i^*) = 1$ to rewrite $\mathbf{d}^\top = \mathbf{c}^\top \mathbf{H}_{S,i^*}\mathbf{G}^{-1}(\mathbf{P}) + \widetilde{\mathbf{e}}^\top + \mathsf{tg}$ up to a noise term that $\widetilde{\mathbf{e}}$ smudges, so that $\mathbf{s}$ now appears only in $\mathbf{c}^\top$,

(4) Invoke LWE with $\mathbf{A}$ set as $\mathbf{M} + \mathsf{encvec}(i^*) \otimes \mathbf{G}$ for an LWE matrix $\mathbf{M}$ to make $\mathbf{c}^\top$ uniform,

(5) Sample tg uniformly so that $\mathbf{d}^\top$ and h become uniform, and,

(6) Apply blindness, so that $\widetilde{C}^*$ is uniform, hence $T^* \oplus \widetilde{C}^*$ is uniform, hence by SEPRF security key is distributed as a fresh key.

At this point the challenge ciphertext is distributed as $\mathsf{pkEnc}(i^*)$ and the key as an honest key. Throughout, the oracle queries for indices $i \neq i^*$ are answered with the real garbling key and the equivocation key, since their PRF inputs differ from t^* with overwhelming probability, and the programming of $\mathbf{A}$ in step (4) is selective in i^* only, which the attacker declares at the start.

Payload hiding. Let $i^* \notin S$ and let μ be adversarial. Only steps (3)-(4) change from above as follows: now $f_S(i^*) = 0$, so $\mathbf{c}^\top \mathbf{H}_{S,i^*}\mathbf{G}^{-1}(\mathbf{P}) \approx \mathbf{s}^\top \mathbf{A}_S \mathbf{G}^{-1}(\mathbf{P})$ and we can rewrite $\mathbf{d}^\top =$

$\mathbf{c}^\top \mathbf{H}_{S,i^*} \mathbf{G}^{-1}(\mathbf{P}) - (\mathbf{s}^\top \mathbf{P} + \mathsf{err}) + \widetilde{\mathbf{e}}^\top + \mathsf{tg}$. The pair $(\mathbf{c}^\top, \mathbf{s}^\top \mathbf{P} + \mathsf{err})$ is an LWE instance with public matrix $[\mathbf{A} - \mathsf{encvec}(i^*) \otimes \mathbf{G} \mid \mathbf{P}]$, so both components become uniform. Then $\mathbf{d}^\top$ hides tg, so $h = \lceil \mathsf{tg} \rfloor_2 \oplus \mu$ is statistically close to uniform even though μ is adversarial. The remaining steps are as before. Full proofs are in Section 4.

A variant for polynomially many indices. The homomorphic evaluation of f_S is the most expensive step. When N is polynomial, we can replace f_S by the inner product with the indicator vector of S, so that $\mathbf{A}_S = \sum_{j \in S} \mathbf{A}_j$ is a sum of blocks of $\mathbf{A}$, and keep the encoding $\mathbf{s}^\top(\mathbf{A} - \mathbb{1}_{N,i}^\top \otimes \mathbf{G})$ ($\mathbb{1}_{N,i}$ is the canonical i-th basis vector) compact by setting $\mathbf{A} = -\mathbf{B}_1 \mathbf{V}_N$ with the matrix commitment scheme of [Wee25]. This variant retains full compactness and public-coin crs under the $2m^2$-decomposed LWE assumption [AMR25], but some running times grow with N, so it cannot replace our main construction in the FBE compiler, where indices are λ-bit tags.

2.4 Application I: Flexible Broadcast Encryption in the Standard Model

An FBE scheme [FWW23] consists of a public-coin global setup GSetup, a key generation algorithm KGen that every user runs on its own to obtain $(\mathsf{pk}, \mathsf{sk})$, an encryption algorithm that takes an arbitrary list of public keys $(\mathsf{pk}_1, \ldots, \mathsf{pk}_N)$, and a decryption algorithm. There is no bound on the number of users and no slot structure. In the adaptive security game, the attacker asks for honestly generated public keys, corrupts any of them, and finally names a challenge list of uncorrupted keys. Semi-static security is the same game without corruptions (full definition in Section 5).

The two-key compiler and its selector bits. Gentry and Waters [GW09] showed how to boost semi-static security to adaptive security, and the same compiler applies to FBE [FWW23]. Every user samples two key pairs, keeps only $\mathsf{fbe.sk}_s$ for a random bit s, and publishes both public keys. To encrypt to N users, the encryptor picks selector bits $t_1, \ldots, t_N$ and encrypts the message twice, once to $(\mathsf{fbe.pk}_{i,t_i})_i$ and once to $(\mathsf{fbe.pk}_{i,1\oplus t_i})_i$. A reduction to semi-static security knows $\mathsf{fbe.sk}_{i,s_i}$ for every user and can answer corruptions, while it arranges $t_i = 1 \oplus s_i$ for the challenge users so that the first ciphertext is encrypted to keys it does not know, which the semi-static challenger supplies. The attacker sees s_i for corrupted users and t_i for challenge users, never both, so the correlation is invisible. The selector bits form an N-bit string that must be compressed into the ciphertext. [GW09] derive them from a random oracle applied to a short seed, which the reduction programs, and [HWW25] replaces the random oracle by a PSPRG whose public parameters sit in the crs while the ciphertext carries a projected key for the recipient set. The public parameters of a PSPRG contain one component per index and are generated at setup together with the seed, so the compiled scheme needs a bound N at setup and a crs of size linear in N. This suffices for distributed broadcast encryption. However, for FBE, as number of users are unbounded, the same compiler is not sufficient.

Our compiler. We replace the N public components by one component per user, sampled by the user itself. The global setup additionally samples $\mathsf{pps.crs} \leftarrow \mathsf{PPS.GSetup}(1^\lambda, 2^\lambda)$, which in our construction is a uniformly random matrix of size $\mathsf{poly}(\lambda)$. A user samples, besides its two semi-static key pairs and its bit s, a random tag $\tau \leftarrow \{0,1\}^\lambda$ and $\mathsf{pps.ct} \leftarrow \mathsf{PPS.pkEnc}(\tau)$, which is a uniformly random string, and publishes $\mathsf{pk} = (\mathsf{fbe.pk}_0, \mathsf{fbe.pk}_1, \tau, \mathsf{pps.ct})$. To encrypt to public keys with tags $S = \{\tau_i\}_i$, the encryptor samples a fresh $\mathsf{pps.msk}$ and $\mathsf{pps.sk}_S \leftarrow \mathsf{PPS.KGen}(\mathsf{pps.msk}, S)$, sets $t_i := \mathsf{PPS.Decode}(S, \mathsf{pps.sk}_S, \tau_i, \mathsf{pps.ct}_i)$. It then encrypts the same message under $(\mathsf{fbe.pk}_{i,t_i})_i$ and $(\mathsf{fbe.pk}_{i,1\oplus t_i})_i$, and outputs $(\mathsf{pps.sk}_S, \mathsf{fbe.ct}_0, \mathsf{fbe.ct}_1)$. Decryption recomputes the t_i from $\mathsf{pps.sk}_S$, so the selector bits are a deterministic function of the ciphertext and the public keys, as in

[GW09, HWW25]. Tags τ for each user collide only with negligible probability. The crs grows only by uniformly random $\mathsf{poly}(\lambda)$ bits. Each public key grows by a tag and one PPS ciphertext. FBE ciphertexts additionally carry one PPS secret key. Observe that this transformation does not need any bound on the number of users, and transparent setup can be preserved if applicable.

Security. We first switch every user's $\mathsf{pps.ct}_j$ from $\mathsf{pkEnc}(\tau_j)$ to $\mathsf{skEnc}(\mathsf{pps.msk}, \tau_j, 1 \oplus s_j)$, one user at a time, with $\mathsf{pps.msk}$ sampled at setup. Once every ciphertext is programmed, the selector bits satisfy $t_i = 1 \oplus s_i$ for all recipients, and the reduction to semi-static security proceeds as in [GW09, FWW23]: it generates $\mathsf{fbe.sk}_{j,s_j}$ itself, obtains $\mathsf{fbe.pk}_{j,1\oplus s_j}$ from the semi-static challenger, and receives $\mathsf{fbe.ct}_0$ as the challenge. Afterwards we switch the ciphertexts back to public mode, then to $\mathsf{skEnc}(\mathsf{pps.msk}, \tau_j, s_j)$, and repeat the argument for $\mathsf{fbe.ct}_1$.

The difficulty lies in the switching step. When the reduction answers the j-th key generation query, it does not know whether this key will end up in the challenge set or not. These two cases call for different PPS properties. If the key ends up in the challenge set, it is never corrupted, so s_j is never revealed, and mode indistinguishability applies with a random message μ, after which we define $s_j := 1 \oplus \mu$. If the key ends up outside the challenge set, the attacker may corrupt it and learn s_j, so the programmed message $1 \oplus s_j$ is adversarial from the point of view of the PPS, and payload hiding applies, since $\tau_j \notin S$ and the reduction can still obtain $\mathsf{pps.sk}_S$. Guessing the membership of all keys at once loses a factor of at most 2^Q for Q users. Instead, we use a piecewise guessing argument [HW09, FKPR14, FJP15, HJO+16, JW16, JKK+17, KW20]. For the j-th switch, the reduction guesses only whether the j-th key lands in the challenge set, and the two branches end in the same hybrid where we sample $\mathsf{pps.ct}_j \leftarrow \mathsf{PPS.skEnc}(\mathsf{pps.msk}, \tau_j, \cdot)$. So each switch costs a constant factor and the overall loss is linear in the number of keys. Full analysis is in Section 5.

Instantiations. Goyal and Yadugiri [GY26a] and Abram, Malavolta, and Roy [AMR26] recently constructed adaptively secure FBE schemes from decomposed LWE in the random oracle model in which every parameter is $\mathsf{poly}(\lambda)$. Along the way, both works design a semi-statically secure FBE scheme from decomposed LWE with the same parameters, and apply the Gentry-Waters compiler with a random oracle. The semi-statically secure scheme of [GY26a] itself relies on a random oracle whereas that of [AMR26] is in the standard model. Plugging the latter into our compiler gives an adaptively secure FBE scheme in the standard model from decomposed LWE with every parameter $\mathsf{poly}(\lambda)$ (Corollary 5.21). We believe that it is possible to remove the random oracles in the design of semi-statically secure FBE scheme of [GY26a] while preserving compactness. We do not pursue this here as our goal is a compactness preserving compiler from semi-static to adaptive security in the standard model that applies to any semi-statically secure scheme. For distributed broadcast encryption, where user i publishes $\mathsf{pps.ct}_i \leftarrow \mathsf{pkEnc}(i)$ for its slot, the same compiler adds only $\mathsf{poly}(\lambda, \log N)$ to the crs. Our PPS also slots into the standard-model equivocal DBE construction of [GY26c], where it removes the 1-key LWE assumption and yields adaptively secure DBE with a crs of size $N \cdot \mathsf{poly}(\lambda, \log N)$ from $2m^2$-decomposed LWE (Corollary 5.22), matching the standard-model DBE result of [AMR26] through matrix commitments [Wee25, WW25] rather than tree encodings.

2.5 Application II: Projective PRGs and Secret Sharing for Graphs

From PPS to a publicly sampleable PPRG. We described the construction in Section 2.1. Adaptive pseudorandomness follows from payload hiding, as a PPS ciphertext hides the output at an index outside the projected set. For sampling indistinguishability, we switch every ciphertext

to public-key mode, one index at a time. The reduction guesses whether the index belongs to the eventual projected set, using mode indistinguishability if it does and payload hiding otherwise. Once all ciphertexts are in public-key mode, the master secret key is unused. Thus, we can generate the projected key under a fresh master secret key, as in Samp. Switching the ciphertexts back under the original master secret key only needs mode indistinguishability, as the attacker holds no key under it (full details in Section 6.1).

Locality. Evaluating the i-th output reads only its PPS ciphertext, which consists of $\mathsf{poly}(\lambda, \log \ell)$ bits of the public parameters for output length ℓ, independent of $|S|$. In comparison, the DDH, bilinear, and LWE based PPRGs of [ABI+23] and the LWE based PSPRG of [HWW25] combine the key with the public values of every index in S. Locality lets us give each party only the public values it needs and avoid a public share.

Secret sharing for graphs. For a graph G on L vertices, [ABI+23] gives vertex v the share $(s \oplus b_v, \mathsf{sk}_{N(v)})$, where s is the secret, b_v is the v-th PPRG output, and $\mathsf{sk}_{N(v)}$ is the projected key for its neighbors. Along an edge $\{u, v\}$, the key of v recovers b_u and hence s. An independent set I only holds keys for sets disjoint from I. Thus, robustness against $|I|$ keys hides $(b_v)_{v \in I}$ and the secret.

We first need robustness against as many keys as the independence number Q of G, whereas PPS only guarantees security against one key per master secret key. In order to argue robustness against an a-priori bounded Q number of secret keys, we use techniques from the bounded collusion functional encryption literature [GVW12, AV19]. In more detail, set $R = \lambda^2(\lambda+1)^2$ and arrange $Q \cdot R$ independent PPS instances into Q buckets of R instances each. In every bucket, share each output $\mu_i \in \mathbb{F}$ using an independent $(\lambda+1)$-out-of-R Shamir sharing [Sha79], and encrypt its shares at index i under the corresponding instances. To generate a projected key for S, choose a random bucket $k \leftarrow [Q]$ and a random set $\Gamma \subseteq [R]$ of size $\lambda+1$. The projected key contains PPS keys for S under the instances (k, r) for $r \in \Gamma$. These keys recover enough Shamir shares to reconstruct any output in S. With overwhelming probability, a load balancing bound [AV19] gives at most λ keys per bucket, and a small pairwise intersection bound [GVW12] gives at most λ instances per bucket used by multiple keys. For a target index outside all the projected sets, every other instance sees at most one key. Thus, payload hiding lets us switch its target ciphertext to public-key mode. The remaining Shamir shares lie below the reconstruction threshold in every bucket and reveal nothing about the target output.

We also need to account for the public parameters. For the LWE based PPRG of [ABI+23], balancing the key size against the public values each vertex needs gives shares of size $L^{1/2} \cdot \mathsf{poly}(\lambda)$ [ABI+23, Corollary 4.2]. By locality, our dealer gives v only the ciphertexts at its index, yielding the share $(s \oplus b_v, (\mathsf{pps.ct}_{v,k,r})_{k,r}, \mathsf{sk}_{N(v)})$ with no public share. Its size is $Q \cdot \mathsf{poly}(\lambda, \log L)$, improving the dependence on L when $Q < \sqrt{L}$ but still growing with Q.

Secret sharing for evolving graphs. Following the evolving graph setting of Francati and Venturi [FV25], we use PPS directly, without first constructing a PPRG. The dealer holds every PPS master secret key and can generate a vertex's ciphertexts and keys independently of each other and of those of other vertices. This allows vertices to join one at a time. The t-th vertex declares its neighbors $S_t \subseteq [t-1]$ among the existing vertices, and the dealer maintains a state and issues a share without modifying existing shares. We define a collusion bound Q on the number of corrupted vertices and a degree bound Δ on $|S_t|$. A scheme supports unbounded collusion or unbounded degree if its parameters grow only logarithmically with the corresponding bound

(Section 6.2).

The dealer sets up $Q \cdot R$ PPS instances with index space $[2^\lambda]$. It gives vertex t encryptions of Shamir shares of s at index t under every instance, together with PPS keys for S_t under $\lambda + 1$ instances of a random bucket. Along an edge $\{u, t\}$ with $u < t$, these keys decrypt enough ciphertexts of u to recover s. The ciphertexts of t refer only to its index, and its keys refer only to S_t. As the index space in PPS is exponential, the dealer needs neither the number of vertices nor a bound on it at setup. Every share has size $Q \cdot \mathsf{poly}(\lambda)$, and the degree is unbounded as PPS keys are independent of the set size. The PPRGs from LWE in [ABI+23, HWW25] fix their output length at setup and publish components for the supported indices. Using PPS directly avoids this setup dependence, although the shares still grow with Q.

Robust PPS. The dependence on Q comes from the single-key security of PPS, which necessitates the bucket construction. To handle arbitrary independent sets without this overhead, we define a generalization and relaxation called robust PPS. We drop mode indistinguishability and allow the public parameters to be sampled together with the master secret key, so that pp may depend on msk. We require payload hiding against any number of keys for sets excluding the target index, interleaved with encryption queries. We retain pkEnc, so PPS gives the corresponding guarantee against one key issued after the encryption queries (full definition is available in Section 6.2).

For evolving graphs, the dealer publishes pp as the public share and gives vertex t the share $(\mathsf{sk}_{S_t}, \mathsf{ct}_t)$, where $\mathsf{sk}_{S_t} \leftarrow \mathsf{KGen}(\mathsf{msk}, S_t)$ and $\mathsf{ct}_t \leftarrow \mathsf{skEnc}(\mathsf{msk}, t, s)$. Along an edge $\{u, t\}$ with $u < t$, $\mathsf{Decode}(S_t, \mathsf{sk}_{S_t}, u, \mathsf{ct}_u)$ recovers s. Both collusion and degree are unbounded, with vertex shares of size $\mathsf{poly}(\lambda)$.

Adaptive security. The attacker chooses arrivals, edges, and corruptions adaptively, provided that the corrupted vertices form an independent set. Adaptive security for evolving access structures was previously studied by Francati, Giammusso, and Venturi [FGV25]. We defer sampling each share until its vertex is corrupted. This preserves the attacker's view, as arrival responses do not depend on shares, which are sampled independently given the master secret key.

We switch $\mathsf{pps.ct}_j$ from an encryption of 0 to an encryption of 1, one vertex j at a time, going through $\mathsf{pkEnc}(j)$. If j is never corrupted, its share is never sampled and the hybrids are identical. If j is corrupted, independence of the corrupted set gives $j \notin S_v$ for every corrupted vertex v. The reduction obtains these keys and the other corrupted vertices' ciphertexts from the robust payload hiding oracle, embedding its challenge as ct_j. This proves adaptive security with a loss linear in the number of vertices, without an additional compiler.

Constructing robust PPS. We obtain robust PPS from public-key encryption schemes whose keys are associated with sets of indices and decrypt exactly the ciphertexts at those indices. Here, $\mathsf{pkEnc}(i)$ simply encrypts 0 at index i under the master public key. Batched IBE [AFP25] provides keys for sets of at most B identities, giving robust PPS for such sets and secret sharing with degree bound $\Delta := B$. The pairing-based instantiations [AFP25, GWWW26] have master public keys of $O(B)$ group elements and keys and ciphertexts of $O(1)$ group elements. The instantiation from ℓ-succinct LWE [XC25] has all three of size $\mathsf{poly}(\lambda, \log B)$ (Corollary 6.24).

Key-policy ABE with succinct keys [BGG+14] removes the degree bound. A key for S is an ABE key for the membership circuit f_S, whose depth is $O(\log N)$ for every $S \subseteq [N]$. A ciphertext at index i uses i as its attribute. Since the key size depends on this depth rather than the circuit size, we obtain robust PPS with no bound on the set size and keys and ciphertexts of size $\mathsf{poly}(\lambda, \log N)$. Taking $N = 2^\lambda$ gives secret sharing for evolving graphs with unbounded collusion and degree.

Shares of size $\mathsf{poly}(\lambda)$ from LWE (Corollary 6.29). In both cases, robust payload hiding follows from selective security with the target index as the challenge identity or attribute, and key queries are admissible exactly when their sets exclude it. [ABI+23, Section 1.1.3] already observed the equivalence between such ABE and secret sharing for bipartite graphs, but used it only for general access structures.

Bounded robust PPS from PPS. Finally, the bucket construction described above in the context of PPRG from PPS gives a bounded variant of robust PPS from PPS alone (Section A). We fix a bound Q on the number of keys at setup. The public parameters contain the $Q \cdot R$ common random strings. A ciphertext at index i encrypts Shamir shares of the message under every instance. A key for S contains PPS keys for S under $\lambda + 1$ instances of a random bucket, together with their common random strings. For a graph on L vertices fixed in advance, using this bounded version with index space $[L]$ of robust PPS gives shares of size $Q \cdot \mathsf{poly}(\lambda, \log L)$ (Corollary A.12).

3 Preliminaries

Notation. By PPT we denote probabilistic polynomial-time. We denote the security parameter by λ and the set of positive integers by $\mathbb{N}$. For any $a, b \in \mathbb{Z}, a \leq b$, we denote by $[a, b]$, the set of all integers from a to b including a and b. In other words, $[a, b] = \{a, \ldots, b\}$. We denote by $[n] := [1, n]$. We denote by $x \leftarrow \mathcal{X}$, the process of sampling an element x from the set $\mathcal{X}$, with uniform probability. Similarly, for any PPT algorithm $\mathcal{A}$, $x \leftarrow \mathcal{A}(y)$ denotes the process of sampling x from the output distribution of $\mathcal{A}$ when run on y with randomly chosen coins. By $\mathsf{negl}(\cdot)$, we denote negligible functions. By $\mathsf{poly}(\cdot)$, we denote positive polynomials. All of our logarithms log are in base 2. Consider two efficiently samplable probability distributions $\mathcal{X} = \{\mathcal{X}_\lambda\}_{\lambda \in \mathbb{N}}$ and $\mathcal{Y} = \{\mathcal{Y}_\lambda\}_{\lambda \in \mathbb{N}}$ over support Ω. We say $\mathcal{X}$ and $\mathcal{Y}$ are computationally indistinguishable if for any non-uniform PPT distinguisher $\mathcal{D} = \{\mathcal{D}_\lambda\}_{\lambda \in \mathbb{N}}$, and for large enough $\lambda \in \mathbb{N}$,

$$\left| \Pr_{\alpha \leftarrow \mathcal{X}_\lambda} \left[1 \leftarrow \mathcal{D}(1^\lambda, \alpha) \right] - \Pr_{\alpha \leftarrow \mathcal{Y}_\lambda} \left[1 \leftarrow \mathcal{D}(1^\lambda, \alpha) \right] \right| \leq \mathsf{negl}(\lambda)$$

We write $\mathcal{X} \approx_c \mathcal{Y}$ to denote that distributions $\mathcal{X}$ and $\mathcal{Y}$ computationally indistinguishable. Similarly, by $\approx_s$, we denote statistical indistinguishability. Concretely, $\mathcal{X} \approx_s \mathcal{Y}$ if for large enough $\lambda \in \mathbb{N}$, $\mathsf{SD}(\mathcal{X}_\lambda, \mathcal{Y}_\lambda) = \frac{1}{2} \sum_{\alpha \in \Omega} |\Pr[\alpha \leftarrow \mathcal{X}_\lambda] - \Pr[\alpha \leftarrow \mathcal{Y}_\lambda]| \leq \mathsf{negl}(\lambda)$. By $\mathcal{X} \equiv \mathcal{Y}$, we denote equivalent distributions.

Definition 3.1 (PRF)**.** A pseudorandom function PRF is a deterministic polynomial-time function that takes inputs from $\mathcal{X} = \{\mathcal{X}_\lambda\}_{\lambda \in \mathbb{N}}$ and a key $K \in \{0,1\}^\lambda$ and outputs values in $\mathcal{R} = \{\mathcal{R}_\lambda\}_{\lambda \in \mathbb{N}}$. We say that the PRF is a secure pseudorandom function if it satisfies the following properties.

Pseudorandomness. For any stateful PPT adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that $\forall\ \lambda \in \mathbb{N}$,

$$\left| \Pr \left[\ b \leftarrow \mathcal{A}^{\mathcal{O}_b(\cdot)}(1^\lambda) \quad : \quad b \xleftarrow{\$} \{0,1\}, K \xleftarrow{\$} \{0,1\}^\lambda \ \right] - \frac{1}{2} \right| \leq \mathsf{negl}(\lambda)$$

where $\mathcal{O}_0(\cdot) = \mathsf{PRF}(K, \cdot)$ and $\mathcal{O}_1(\cdot)$ generates consistent random strings in $\mathcal{R}_\lambda$.

3.1 Lattice Preliminaries

For prime power $q \in \mathbb{N}$, we denote the finite field with q elements as $\mathbb{Z}_q$. Following popular convention, we denote column vectors by lowercase bold letters, e.g., $\mathbf{x} \in \mathbb{Z}_q^n$ and denote matrices by uppercase bold letters, e.g., $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$. We use x_i to denote the i-th element in $\mathbf{x}$ for $i \in [n]$ and $a_{i,j}$ to denote the element in the i-th row and j-th column of matrix $\mathbf{A}$ for $i \in [n], j \in [m]$. We define $\mathbf{1}_n$ to be a vector of all ones, $\mathbf{0}_n$ to be a vector of all zeros, and $\mathbb{1}_{n,j}$ to denote the n-dimensional canonical basis vector that has 1 in the j-th position, i.e., $\mathbf{1}_n = [1\ 1\ \ldots\ 1]^\top$ and $\mathbb{1}_{n,j} = [\ \ldots\ 0\ 1\ 0\ \ldots]^\top$. We frequently drop n from subscript in these vectors when dimension is clear. We denote the $n \times n$ identity matrix by $\mathbf{I}_n$ and drop the subscript when dimension is clear. For a matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$, $\mathsf{vec}(\mathbf{A}) \in \mathbb{Z}_q^{nm}$ denoted the vector obtained by concatenating the columns of $\mathbf{A}$. For any vector $\mathbf{a} \in \mathbb{Z}_q^n$ or matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$, we use $\|\mathbf{a}\|$ and $\|\mathbf{A}\|$ to denote the maximum of all the entries in $\mathbf{a}$ and $\mathbf{A}$ respectively. We use $\mathbf{A} \otimes \mathbf{B}$ to denote the Kronecker product between matrices $\mathbf{A}, \mathbf{B}$. The rounding function $\lceil \cdot \rfloor_p : \mathbb{Z}_q \to \mathbb{Z}_p$ is given by $\lceil x \rfloor_p = \lceil \frac{p}{q} x \rfloor \in \mathbb{Z}_p$. That is, if $x = y \cdot q/p + z$ where $z \in [-q/2p, q/2p)$, we have $\lceil x \rfloor_p = y$. The rounding function can be extended to vectors by considering coordinate-wise rounding.

Discrete Gaussian. We write $\mathcal{D}_{\mathbb{Z},\sigma}$ to denote the discrete gaussian distribution with width parameter $\sigma > 0$. We state the following important lemma about the maximum norm of a value sampled from discrete gaussian distribution.

Lemma 3.2 ([Lyu12, MP12])**.** For any $k > 0$, $\Pr[|x| > k\sigma : x \leftarrow \mathcal{D}_{\mathbb{Z},\sigma}] \leq 2\exp\left(\frac{-k^2}{2}\right)$.

Gadget Matrix. For positive integers n, m, q, and $m = n\lceil \log q \rceil$, we write $\mathbf{G} \in \mathbb{Z}_q^{n \times m}$ to denote the gadget matrix [MP12]. It is defined as $\mathbf{G} = (\mathbf{I}_n \otimes \mathbf{g}^\top)$, $\mathbf{g} = [1\ 2\ 2^2\ \ldots\ 2^{\lceil \log q \rceil - 1}]^\top \in \mathbb{Z}_q^{\lceil \log q \rceil}$. We write $\mathbf{G}^{-1}$ to denote the standard entry-wise bit decomposition. Note that $\mathbf{G}^{-1} : \mathbb{Z}_q^n \to \mathbb{Z}_q^m$ is an operation such that $\mathbf{G}\mathbf{G}^{-1}(\mathbf{a}) = \mathbf{a}$ and not inverse of $\mathbf{G}$. Output of $\mathbf{G}^{-1}$ is padded with 0s to match the dimensions. We often overload the notation and write $\mathbf{G}^{-1}(\mathbf{A})$ where $\mathbf{G}^{-1}$ is performed for each column.

Lemma 3.3 (Smudging Lemma [AJL+12])**.** Let $B_1 = B_1(\lambda)$ and $B_2 = B_2(\lambda)$ be positive integers and let $e_1 \in [-B_1, B_1]$ be a fixed integer. Let $e_2 \leftarrow [-B_2, B_2]$ be chosen uniformly at random. Then the distribution of e_2 is statistically indistinguishable from that of $e_2 + e_1$ as long as $B_1/B_2 = \mathsf{negl}(\lambda)$.

Function Class $\mathcal{F}_{d,\ell}$. The ensemble function class $\mathcal{F}_{d,\ell} = \{\mathcal{F}_{d,\ell}\}_{d \in \mathbb{N}, \ell \in \mathbb{N}}$ consists of the set of all functions that take ℓ-bit inputs and can be computed by a depth-d boolean circuit.

Homomorphic Computation. Let (n, m, q) be LWE parameters such that $m \geq n\lceil \log q \rceil$. The lattice homomorphic computation machinery from [GSW13, BGG+14] for function class $\mathcal{F}_{d,\ell}$ provides two *deterministic* polynomial-time algorithms described as follows.

$\mathsf{EvalF}(\mathbf{B}, f) \to \mathbf{B}_f$**.** The function evaluation algorithm takes as input $\mathbf{B} \in \mathbb{Z}_q^{n \times \ell m}$, function $f \in \mathcal{F}_{d,\ell}$, and outputs matrix $\mathbf{B}_f \in \mathbb{Z}_q^{n \times m}$.

$\mathsf{EvalFX}(\mathbf{B}, f, \mathbf{x}) \to \mathbf{H}_{f,\mathbf{x}}$**.** The function-input evaluation algorithm takes as input $\mathbf{B} \in \mathbb{Z}_q^{n \times \ell m}$, function $f \in \mathcal{F}_{d,\ell}$, input $\mathbf{x} \in \{0,1\}^\ell$, and outputs matrix $\mathbf{H}_{f,\mathbf{x}} \in \mathbb{Z}_q^{\ell m \times m}$.

These algorithms satisfy these two properties: for any $\mathbf{B}$, $f \in \mathcal{F}_{d,\ell}, \mathbf{x} \in \{0,1\}^\ell$ (1) $(\mathbf{B} - \mathbf{x}^\top \otimes \mathbf{G})\mathbf{H}_{f,\mathbf{x}} = \mathbf{B}_f - f(x)\mathbf{G}$ where $\mathbf{B}_f = \mathsf{EvalF}(\mathbf{B}, f)$, (2) $\|\mathbf{H}_{f,\mathbf{x}}\| \leq m^{O(d)}$. In particular, this is independent of size of f.

Learning with Errors Assumption. We provide the definition of the learning with errors (LWE) assumption.

Assumption 3.4 ([Reg05])**.** The LWE assumption with parameters (n, q, χ) states that

$$\left\{(\mathbf{A}, \mathbf{s}^\top\mathbf{A} + \mathbf{e}^\top)\right\} \approx_c \left\{(\mathbf{A}, \mathbf{c}^\top)\right\}$$

where $m \in \mathbb{N}$, $\mathbf{A} \leftarrow \mathbb{Z}_q^{n\times m}, \mathbf{s} \leftarrow \mathbb{Z}_q^n, \mathbf{e} \leftarrow \mathcal{D}_{\mathbb{Z},\chi}^m, \mathbf{c} \leftarrow \mathbb{Z}_q^m$.

4 Programmable Pseudorandom Set Sampler

In this section, we provide the definition and construction of a programmable pseudorandom set sampler scheme (PPS) adapted from [GY26c]. The main distinctions are that we define a 'stronger' version of security[1] for unbounded indices.

4.1 Definition

Syntax. A programmable pseudorandom set sampler (PPS) scheme for message space $\mathcal{M} = \{\mathcal{M}_\lambda\}_{\lambda\in\mathbb{N}}$ consists of the following polynomial time algorithms.

$\mathsf{GSetup}(1^\lambda, N) \to \mathsf{crs}$**.** The randomized public parameter generation algorithm takes as input security parameter λ, number of indices N (in binary), and outputs common random string crs. The following algorithms takes crs implicitly.

$\mathsf{MSKGen} \to \mathsf{msk}$ The randomized master secret key generation algorithm takes as input common random string crs and outputs master secret key msk.

$\mathsf{pkEnc}(i) \to \mathsf{ct}$**.** The randomized *public-key* encryption algorithm takes as input index $i \in [N]$ and outputs ciphertext ct.

$\mathsf{KGen}(\mathsf{msk}, S) \to \mathsf{sk}_S$**.** The possibly randomized secret key generation algorithm takes as input master secret key msk, set $S \subseteq [N]$, and outputs secret key sk_S.

$\mathsf{skEnc}(\mathsf{msk}, i, \mu) \to \mathsf{ct}$**.** The randomized *secret-key* encryption algorithm takes as input master secret key msk, index $i \in [N]$, message $\mu \in \mathcal{M}_\lambda$, and outputs ciphertext ct.

$\mathsf{Decode}(S, \mathsf{sk}_S, i, \mathsf{ct}) \to \mu'$**.** The deterministic decoding algorithm takes as input a ciphertext ct, secret key sk_S for set S, and outputs $\mu' \in \mathcal{M}_\lambda \cup \{\perp\}$.

Definition 4.1 (PPS)**.** A PPS scheme $(\mathsf{GSetup}, \mathsf{MSKGen}, \mathsf{pkEnc}, \mathsf{KGen}, \mathsf{Decode}, \mathsf{skEnc})$ is said to be a programmable pseudorandom set sampler scheme (PPS) if it satisfies the following properties.

[1] It could be possible that [GY26c] implicitly satisfies this security notion for a bounded number of indices.

Correctness. For any $\lambda \in \mathbb{N}$, $N \leq 2^\lambda$, sets $S \subseteq [N], i \in S$, for any $\mu \in \mathcal{M}$,

$$\Pr\left[\begin{array}{l} \mu = \mathsf{Decode}(S, \mathsf{sk}_S, i, \mathsf{ct}) \end{array} : \begin{array}{l} \mathsf{crs} \leftarrow \mathsf{GSetup}(1^\lambda, N), \\ \mathsf{msk} \leftarrow \mathsf{MSKGen}(\mathsf{crs}), \\ \mathsf{ct} \leftarrow \mathsf{skEnc}(\mathsf{msk}, i, \mu), \\ \mathsf{sk}_S \leftarrow \mathsf{KGen}(\mathsf{msk}, S) \end{array}\right] = 1$$

Full Compactness. There exist polynomials $\mathsf{poly}_{\mathsf{crs}}(\cdot,\cdot), \mathsf{poly}_{\mathsf{msk}}(\cdot,\cdot), \mathsf{poly}_{\mathsf{ct}}(\cdot,\cdot), \mathsf{poly}_{\mathsf{sk}}(\cdot,\cdot)$ such that $\forall\, \lambda, N \leq 2^\lambda, S \subseteq [N], |\mathsf{crs}| = \mathsf{poly}_{\mathsf{crs}}(\lambda, \log N), |\mathsf{msk}| = \mathsf{poly}_{\mathsf{msk}}(\lambda, \log N), |\mathsf{ct}| = \mathsf{poly}_{\mathsf{ct}}(\lambda, \log N)$, $|\mathsf{sk}_S| = \mathsf{poly}_{\mathsf{sk}}(\lambda, \log N)$.

Mode Indistinguishability. For any stateful admissible PPT adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for any $\lambda \in \mathbb{N}$,

$$\left|\Pr\left[\begin{array}{l} b \leftarrow \mathcal{A}(\mathsf{sk}_S) \end{array} : \begin{array}{l} (N, i^* \in [N]) \leftarrow \mathcal{A}(1^\lambda), b \leftarrow \{0,1\}, \\ \mathsf{crs} \leftarrow \mathsf{GSetup}(1^\lambda, N), \mathsf{msk} \leftarrow \mathsf{MSKGen}(\mathsf{crs}), \\ \text{if } b = 0, \mathsf{ct}_{i^*,0} \leftarrow \mathsf{pkEnc}(i^*), \\ \text{if } b = 1, \mu_{i^*} \leftarrow \mathcal{M}_\lambda, \mathsf{ct}_{i^*,1} \leftarrow \mathsf{skEnc}(\mathsf{msk}, i^*, \mu_{i^*}), \\ S \leftarrow \mathcal{A}^{\mathcal{O}(\cdot,\cdot)}(\mathsf{crs}, \mathsf{ct}_{i^*,b}), \\ \mathsf{sk}_S \leftarrow \mathsf{KGen}(\mathsf{msk}, S) \end{array}\right] - \frac{1}{2}\right| \leq \mathsf{negl}(\lambda)$$

where the oracle $\mathcal{O}$ takes as input $(i, \mu_i) \in [N] \times \mathcal{M}_\lambda$, $i \neq i^*$ and responds with $\mathsf{ct}_i \leftarrow \mathsf{skEnc}(\mathsf{msk}, i, \mu)$. A stateful PPT machine $\mathcal{A}$ is said to be admissible if $S \subseteq [N]$ and $i^* \in S$.

Payload Hiding. For any stateful admissible PPT adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for any $\lambda \in \mathbb{N}$,

$$\left|\Pr\left[\begin{array}{l} b \leftarrow \mathcal{A}(\mathsf{sk}_S) \end{array} : \begin{array}{l} (N, i^* \in [N], \mu_{i^*} \in \mathcal{M}_\lambda) \leftarrow \mathcal{A}(1^\lambda), b \leftarrow \{0,1\}, \\ \mathsf{crs} \leftarrow \mathsf{GSetup}(1^\lambda, N), \mathsf{msk} \leftarrow \mathsf{MSKGen}(\mathsf{crs}), \\ \text{if } b = 0, \mathsf{ct}_{i^*,0} \leftarrow \mathsf{pkEnc}(i^*), \\ \text{if } b = 1, \mathsf{ct}_{i^*,1} \leftarrow \mathsf{skEnc}(\mathsf{msk}, i^*, \mu_{i^*}), \\ S \leftarrow \mathcal{A}^{\mathcal{O}(\cdot,\cdot)}(\mathsf{crs}, \mathsf{ct}_{i^*,b}), \\ \mathsf{sk}_S \leftarrow \mathsf{KGen}(\mathsf{msk}, S) \end{array}\right] - \frac{1}{2}\right| \leq \mathsf{negl}(\lambda)$$

where the oracle $\mathcal{O}$ takes as input $(i, \mu_i) \in [N] \times \mathcal{M}_\lambda$, $i \neq i^*$ and responds with $\mathsf{ct}_i \leftarrow \mathsf{skEnc}(\mathsf{msk}, i, \mu)$. A stateful PPT machine $\mathcal{A}$ is said to be admissible if $S \subseteq [N]$ and $i^* \notin S$.

4.2 Preliminaries

We provide the definitions of two objects we use in our construction.

Somewhere Equivocal Pseudorandom Functions. We provide the definition of a somewhere equivocal PRF scheme (SEPRF). Essentially, an SEPRF is a pseudorandom function with an alternate way of generating a key that equivocates the output at a special x^* to a random value y^* of our choice. A somewhere equivocal PRF (SEPRF) for input size λ consists of the following polynomial-time algorithms.

$\mathsf{Setup}(1^\lambda, 1^{\ell_{\mathsf{out}}}) \to \mathsf{key}$**.** The probabilistic setup algorithm takes as input security parameter λ, output size ℓ_{out}, and outputs key key.

$\mathsf{Eval}(\mathsf{key}, x) \to y$**.** The deterministic evaluation algorithm takes as input key key, input $x \in \{0,1\}^\lambda$, and outputs value $y \in \{0,1\}^\ell$.

$\mathsf{Sim}_1(1^\lambda, 1^{\ell_{\mathsf{out}}}, x^*) \to \mathsf{eqKey}$**.** The probabilistic equivocal key generation algorithm takes as input security parameter λ, output size ℓ_{out}, input $x^* \in \{0,1\}^\lambda$ and outputs equivocation key eqKey.

$\mathsf{Sim}_2(\mathsf{eqKey}, y^*) \to \mathsf{key}$**.** The possibly randomized simulated key generation algorithm takes as input equivocation key eqKey, output $y^* \in \{0,1\}^{\ell_{\mathsf{out}}}$, and outputs key key.

Definition 4.2. An SEPRF scheme $(\mathsf{Setup}, \mathsf{Sim}_1, \mathsf{Sim}_2, \mathsf{Eval})$ is said to be a somewhere equivocal PRF scheme (SEPRF) if it satisfies the following properties.

Correctness. For any $\lambda \in \mathbb{N}, \ell_{\mathsf{out}} = \ell_{\mathsf{out}}(\lambda), x^* \in \{0,1\}^\lambda, y^* \in \{0,1\}^{\ell_{\mathsf{out}}}$, and for any $x \neq x^*$, $\mathsf{Eval}(\mathsf{key}, x) = \mathsf{Eval}(\mathsf{eqKey}, x)$. In addition, $\mathsf{Eval}(\mathsf{key}, x^*) = y^*$.

Here, $\mathsf{eqKey} \leftarrow \mathsf{Sim}_1(1^\lambda, 1^{\ell_{\mathsf{out}}}, x^*)$ and $\mathsf{key} \leftarrow \mathsf{Sim}_2(\mathsf{eqKey}, y^*)$.

Security. For any stateful PPT adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that $\forall\ \lambda \in \mathbb{N}$,

$$\left| \Pr \left[b \leftarrow \mathcal{A}(\mathsf{key}_b) \quad : \quad \begin{array}{l} b \xleftarrow{\$} \{0,1\}, (1^{\ell_{\mathsf{out}}}, x^*) \leftarrow \mathcal{A}(1^\lambda), \\ \text{if } b = 0, \mathsf{key}_0 \leftarrow \mathsf{Setup}(1^\lambda, 1^{\ell_{\mathsf{out}}}), \\ \text{if } b = 1, \mathsf{eqKey} \leftarrow \mathsf{Sim}_1(1^\lambda, 1^{\ell_{\mathsf{out}}}, x^*), \\ y^* \xleftarrow{\$} \{0,1\}^{\ell_{\mathsf{out}}}, \mathsf{key}_1 \leftarrow \mathsf{Sim}_2(\mathsf{eqKey}, y^*) \end{array} \right] - \frac{1}{2} \right| \leq \mathsf{negl}(\lambda)$$

Theorem 4.3 ([HJO+16])**.** Assuming the existence of one-way functions, there exists an SEPRF scheme for any positive polynomial $\ell_{\mathsf{out}}(\lambda)$ with $|\mathsf{key}| = O(\lambda^2 \cdot \ell_{\mathsf{out}})$.

Blind Garbled Circuits. A blind garbled circuit scheme is a strengthening of a garbled circuit scheme with the additional property that the simulated garbled circuit is a uniformly random string as long as the circuit's output is also uniformly random. We also employ the reusable property as provide in [WW24]. A blind garbled circuit scheme (Garb) for a class of circuit $\mathcal{C} = \{\mathcal{C}_{s,\ell}\}_{s \in \mathbb{N}, \ell \in \mathbb{N}}$ consists of the following polynomial-time algorithms.

$\mathsf{Setup}(1^\lambda, 1^s, 1^\ell) \to \mathsf{gk}$**.** The probabilistic setup algorithm takes as input security parameter λ, maximum circuit size s, and input size ℓ, outputs secret garbling key gk.

$\mathsf{InpGarb}(\mathsf{gk}, x) \to \widetilde{x}$**.** The deterministic input garbling algorithm takes as input secret garbling key gk, circuit $x \in \{0,1\}^\ell$ output garbled input $\widetilde{x}$.

$\mathsf{CircGarb}(\mathsf{gk}, C) \to \widetilde{C}$**.** The probabilistic circuit garbling algorithm takes as input secret garbling key gk, circuit $C \in \mathcal{C}_{s,\ell}$, and outputs garbled circuit $\widetilde{C}$.

$\mathsf{Eval}(\widetilde{C}, \widetilde{x}) \to y$**.** The deterministic evaluation algorithm takes as input garbled circuit $\widetilde{C}$, garbled input $\widetilde{x}$, and outputs value y.

$\mathsf{Sim}(\mathsf{gk}, x, C(x)) \to \widetilde{C}$**.** The probabilistic circuit garbling simulation algorithm takes as input secret garbling key gk, input x, output value $C(x)$, and outputs garbled circuit $\widetilde{C}$.

Definition 4.4. A Garb scheme $(\mathsf{Setup}, \mathsf{InpGarb}, \mathsf{CircGarb}, \mathsf{Eval}, \mathsf{Sim})$ is said to be a Garb scheme for circuit class $\mathcal{C} = \{\mathcal{C}_{s,\ell}\}_{s\in\mathbb{N},\ell\in\mathbb{N}}$, if it satisfies the following properties.

Correctness. For any $\lambda \in \mathbb{N}, s = s(\lambda), \ell = \ell(\lambda)$, $C \in \mathcal{C}_{s,\ell}, x \in \{0,1\}^\ell$, $C(x) = \mathsf{Eval}(\widetilde{C}, \widetilde{x})$ where $\widetilde{C} \leftarrow \mathsf{CircGarb}(\mathsf{gk}, C)$, $\widetilde{x} \leftarrow \mathsf{InpGarb}(\mathsf{gk}, x)$, and $\mathsf{gk} \leftarrow \mathsf{Setup}(1^\lambda, 1^s, 1^\ell)$.

Security. For any stateful admissible PPT adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for any $\lambda \in \mathbb{N}$,

$$\left|\Pr\left[b \leftarrow \mathcal{A}^{\mathsf{CircGarb}(\mathsf{gk},\cdot)}(\widetilde{C}) \quad : \quad \begin{array}{l} b \leftarrow \{0,1\}, (1^s, 1^\ell) \leftarrow \mathcal{A}(1^\lambda), \\ \mathsf{gk} \leftarrow \mathsf{Setup}(1^\lambda, 1^s, 1^\ell), \\ x \leftarrow \mathcal{A}^{\mathsf{CircGarb}(\mathsf{gk},\cdot)}(1^\lambda), \\ \widetilde{x} \leftarrow \mathsf{InpGarb}(\mathsf{gk}, x), \\ C \leftarrow \mathcal{A}^{\mathsf{CircGarb}(\mathsf{gk},\cdot)}(\widetilde{x}), \\ \text{if } b = 0, \widetilde{C} \leftarrow \mathsf{CircGarb}(\mathsf{gk}, C), \\ \text{if } b = 1, \widetilde{C} \leftarrow \mathsf{Sim}(\mathsf{gk}, x, C(x)) \end{array}\right] - \frac{1}{2}\right| \leq \mathsf{negl}(\lambda)$$

A stateful PPT machine $\mathcal{A}$ is said to be admissible if it makes at most one challenge input garbling and challenge circuit garbling queries.

Blindness. For any $\lambda \in \mathbb{N}, s = s(\lambda), \ell = \ell(\lambda)$, $C \in \mathcal{C}_{s,\ell}, x \in \{0,1\}^\ell$,

$$\left\{\widetilde{C} \ : \ y \xleftarrow{\$} \{0,1\}^{\ell_{\mathsf{out}}}, \widetilde{C} \leftarrow \mathsf{Sim}(\mathsf{gk}, x, y)\right\} \equiv \left\{\widetilde{C} \ : \ \widetilde{C} \xleftarrow{\$} \{0,1\}^L\right\}$$

where $\mathsf{gk} \leftarrow \mathsf{Setup}(1^\lambda, 1^s, 1^\ell)$. L is the length of the garbled circuit of C and ℓ_{out} is the output length of C.

Theorem 4.5 ([BLSV18, WW24])**.** Assuming the existence of one-way functions, there exists a blind garbled circuit scheme for any positive polynomial s, ℓ with $|\mathsf{gk}| = \ell(2\lambda+1), |\widetilde{x}| = \ell(\lambda+1)$.

4.3 Construction

We now construct a PPS scheme with full compactness (Definition 4.1) from LWE.

Notation. We will use the notation $\mathsf{encvec}(i)$ to write an index $i \in [N]$ as a canonical bit vector $\mathbf{i}^\top \in \{0,1\}^{1\times\lceil\log N\rceil}$. That is, the helper function $\mathsf{encvec}(\cdot)$ encodes an index into a vector.

Parameters. We provide the LWE parameters we use in our construction below.

- $n = \mathsf{poly}(\lambda, \log N)$, $\chi = \mathsf{poly}(n)$ (LWE dimensions)
- $m = n \cdot \lceil \log q \rceil$ (Gadget matrix)
- $\widetilde{\chi} > \lambda^{\omega(1)} \cdot \chi \cdot \lceil \log N \rceil \cdot m^{\mathsf{polylog}(N)}$ (Error smudging)
- $B = \chi \cdot \widetilde{\chi} \cdot \lceil \log N \rceil \cdot m^{\mathsf{polylog}(N)}$ (Accumulated error)
- $q > \lambda^{\omega(1)} \cdot \lambda \cdot (4B+2)$ (Correctness)

One way to set these parameters is $n = (\lambda \cdot \mathsf{polylog}(N))^{3/\epsilon} \cdot \mathsf{polylog}(\lambda, N), \chi = n$, q to be a prime of size $2^{\mathsf{poly}(\lambda, \log N)}$, and $m = n\lceil \log q \rceil$. This will yield $q/\chi = 2^{\widetilde{O}(n^\epsilon)}$ for any $\epsilon < 1/2$ which would require LWE with sub-exponential modulus-to-noise ratio.

Construction 4.6. We provide the construction of a PPS scheme for $\mathcal{M}_\lambda = \{0,1\}^\lambda$ using an SEPRF scheme (Definition 4.2) and a blind garble circuit scheme Garb (Definition 4.4), and relying on the LWE assumption (Assumption 3.4) below.

$\mathsf{GSetup}(1^\lambda, N)$**.** Sample matrix $\mathbf{A} \leftarrow \mathbb{Z}_q^{n \times (\lceil \log N \rceil \cdot m)}$. Output $\mathsf{crs} := \mathbf{A}$.

MSKGen**.** Sample $\mathsf{prf.key} \leftarrow \mathsf{SEPRF.Setup}(1^\lambda, 1^{L_{\tilde{C}}})$ and $\mathsf{garb.gk} \leftarrow \mathsf{Garb.Setup}(1^\lambda, 1^{s_C}, 1^{n \cdot m \cdot \lceil \log q \rceil})$. Here, s_C is the size of circuit from Figure 1 and $L_{\widetilde{C}}$ is the length of the garbled circuit resulted from a blind garbling scheme. Output $\mathsf{msk} := (\mathsf{prf.key}, \mathsf{garb.gk})$.

$\mathsf{pkEnc}(i)$**.** Given $i \in [N]$, sample $t \leftarrow \{0,1\}^\lambda$, $T \leftarrow \{0,1\}^{L_{\tilde{C}}}$. Output $\mathsf{ct} := (t, T)$.

$\mathsf{KGen}(\mathsf{msk}, S)$**.** Parse msk as $(\mathsf{prf.key}, \mathsf{garb.gk})$. Do the following.

1. Given $S \subseteq [N]$, consider the circuit f_S where $f_S(x) = 1$ if $x \in S$ and 0 otherwise. This is a set membership function.
2. Compute $\mathbf{A}_S = \mathsf{EvalF}(\mathbf{A}, f_S)$. Compute $\widetilde{\mathbf{A}}_S \leftarrow \mathsf{Garb.InpGarb}(\mathsf{garb.gk}, \mathbf{A}_S)$.

Output $\mathsf{sk}_S := (\mathsf{prf.key}, \widetilde{\mathbf{A}}_S)$.

$\mathsf{skEnc}(\mathsf{msk}, i, \mu)$**.** Parse msk as $(\mathsf{prf.key}, \mathsf{garb.gk})$. Do the following.

1. Sample $\mathbf{s} \leftarrow \mathbb{Z}_q^n, \mathbf{e} \leftarrow \mathcal{D}_{\mathbb{Z},\chi}^{\lceil \log N \rceil \cdot m}, \mathbf{P} \leftarrow \mathbb{Z}_q^{n \times \lambda}, \widetilde{\mathbf{e}} \leftarrow [-\sqrt{m}\widetilde{\chi}, \sqrt{m}\widetilde{\chi}]^\lambda$. Also, sample $\mathsf{tg} \leftarrow \mathbb{Z}_q^\lambda$ subject to the constraint that each entry of tg is not inside $[-\lfloor q/4 \rfloor - B, -\lfloor q/4 \rfloor + B] \cup [\lfloor q/4 \rfloor - B, \lfloor q/4 \rfloor + B]$. As $q > \lambda^{\omega(1)} B$, the set of valid tg is non-empty.
2. Construct the circuit C from Figure 1.
3. Sample $\widetilde{C} \leftarrow \mathsf{Garb.CircGarb}(\mathsf{garb.gk}, C)$.
4. Sample $t \leftarrow \{0,1\}^\lambda$ and compute $T = \mathsf{SEPRF.Eval}(\mathsf{prf.key}, t) \oplus \widetilde{C}$.

Output $\mathsf{ct} := (t, T)$.

$\mathsf{Decode}(S, \mathsf{sk}_S, i, \mathsf{ct})$**.** Parse sk_S as $(\mathsf{prf.key}, \widetilde{\mathbf{A}}_S)$, ct as (t, T). If $i \notin S$, output $\perp$. Otherwise, do the following.

1. Compute $\widetilde{C} = T \oplus \mathsf{SEPRF.Eval}(\mathsf{prf.key}, t)$.
2. Compute $(\mathbf{c}^\top, \mathbf{P}, \mathbf{d}^\top, h) = \mathsf{Garb.Eval}(\widetilde{C}, \widetilde{\mathbf{A}}_S)$.
3. Construct the same f_S circuit using S like in Step 1 of KGen. Then, compute $\mathbf{H}_{S,i} = \mathsf{EvalFX}(\mathbf{A}, f_S, \mathsf{encvec}(i))$. Recall, $\mathsf{encvec}(i) \in \{0,1\}^{\lceil \log N \rceil}$ is the representation of i as a bit string.
4. Compute $\mathsf{tg}' = \mathbf{d}^\top - \mathbf{c}^\top \mathbf{H}_{S,i} \mathbf{G}^{-1}(\mathbf{P}) \in \mathbb{Z}_q^\lambda$.

Output $\mu' := h \oplus \lceil \mathsf{tg}' \rfloor_2$.

Circuit C

Input: $\mathbf{b} \in \{0,1\}^{n \cdot m \cdot \lceil \log q \rceil}$.
Hardwired: $\mathbf{A} \in \mathbb{Z}_q^{n \times (\lceil \log N \rceil \cdot m)}$, $\mathbf{s} \in \mathbb{Z}_q^n, \mathbf{e} \in \mathbb{Z}_q^{\lceil \log N \rceil \cdot m}, \mathbf{P} \in \mathbb{Z}_q^{n \times \lambda}, \widetilde{\mathbf{e}} \in \mathbb{Z}_q^\lambda, \mathsf{tg} \in \mathbb{Z}_q^\lambda$. Also, $i \in [N], \mu \in \{0,1\}^\lambda$.
Output: Parse input $\mathbf{b}$ as $\mathbf{B} \in \mathbb{Z}_q^{n \times m}$. Do the following.

1. Compute $\mathbf{c}^\top = \mathbf{s}^\top(\mathbf{A} - \mathsf{encvec}(i) \otimes \mathbf{G}) + \mathbf{e}^\top$.
2. Compute $\mathbf{d}^\top = \mathbf{s}^\top(\mathbf{B}\mathbf{G}^{-1}(\mathbf{P})) - \mathbf{s}^\top\mathbf{P} + \widetilde{\mathbf{e}}^\top + \mathsf{tg}$
3. Compute $h = \lceil \mathsf{tg} \rfloor_2 \oplus \mu$.

Output $(\mathbf{c}^\top, \mathbf{P}, \mathbf{d}^\top, h) \in \mathbb{Z}_q^{1 \times (\lceil \log N \rceil m)} \times \mathbb{Z}_q^{n \times \lambda} \times \mathbb{Z}_q^{1 \times \lambda} \times \{0,1\}^\lambda$. This will be given out as a boolean string using the encoding of Remark 4.9 and parsed in this exact format later.

Figure 1: Description of circuit C

Correctness. Note that if $i \notin S$, Decode outputs $\perp$. Hence, we will only focus on the case when $i \in S$. A secret key sk_S is of the form $(\mathsf{prf.key}, \mathsf{Garb.InpGarb}(\mathsf{garb.gk}, \mathbf{A}_S))$ and ct from skEnc has the form $(t, \mathsf{SEPRF.Eval}(\mathsf{prf.key}, t) \oplus \mathsf{Garb.CircGarb}(\mathsf{garb.gk}, C))$ for the same $\mathsf{prf.key}, \mathsf{garb.gk}$ from msk. Here, $\mathbf{A}_S = \mathsf{EvalF}(\mathbf{A}, f_S)$ and the same f_S is considered in Decode as well.

Thus, in Step 2 of Decode, by correctness of Garb and SEPRF, we get $(\mathbf{c}^\top, \mathbf{P}, \mathbf{d}^\top, h) = (\mathbf{s}^\top(\mathbf{A} - \mathsf{encvec}(i) \otimes \mathbf{G}) + \mathbf{e}^\top, \mathbf{P}, \mathbf{s}^\top(\mathbf{A}_S\mathbf{G}^{-1}(\mathbf{P})) - \mathbf{s}^\top\mathbf{P} + \widetilde{\mathbf{e}} + \mathsf{tg}, \lceil \mathsf{tg} \rfloor_2 \oplus \mu)$ for some $\mathbf{s}, \mathbf{e}, \widetilde{\mathbf{e}}, \mathbf{P}, \mathsf{tg}$ of appropriate dimensions chosen by skEnc. By the relations of matrix homomorphic computation machinery,

$$\begin{aligned}
\mathsf{tg}' &= \mathbf{s}^\top(\mathbf{A}_S\mathbf{G}^{-1}(\mathbf{P})) - \mathbf{s}^\top\mathbf{P} + \widetilde{\mathbf{e}}^\top + \mathsf{tg} - (\mathbf{s}^\top(\mathbf{A} - \mathsf{encvec}(i) \otimes \mathbf{G}) + \mathbf{e}^\top)\mathbf{H}_{S,i}\mathbf{G}^{-1}(\mathbf{P}) \\
&= \mathbf{s}^\top(\mathbf{A}_S\mathbf{G}^{-1}(\mathbf{P})) - \mathbf{s}^\top\mathbf{P} + \widetilde{\mathbf{e}}^\top + \mathsf{tg} - (\mathbf{s}^\top(\mathbf{A}_S - \mathbf{G}) + \mathbf{e}^\top\mathbf{H}_{S,i})\mathbf{G}^{-1}(\mathbf{P}) \\
&= \mathbf{s}^\top(\mathbf{A}_S\mathbf{G}^{-1}(\mathbf{P})) - \mathbf{s}^\top\mathbf{P} + \widetilde{\mathbf{e}}^\top + \mathsf{tg} - (\mathbf{s}^\top(\mathbf{A}_S - \mathbf{G})\mathbf{G}^{-1}(\mathbf{P})) - \mathbf{e}^\top\mathbf{H}_{S,i}\mathbf{G}^{-1}(\mathbf{P}) \\
&= \mathsf{tg} + \widetilde{\mathbf{e}}^\top - \mathbf{e}^\top\mathbf{H}_{S,i}\mathbf{G}^{-1}(\mathbf{P})
\end{aligned}$$

Now, we will bound $\left\|\widetilde{\mathbf{e}} - \mathbf{e}^\top\mathbf{H}_{S,i}\mathbf{G}^{-1}(\mathbf{P})\right\|$. By matrix homomorphic computation machinery, $\|\mathbf{H}_{S,i}\| \leq m^{O(d_f)}$ where d_f is the depth of f_S. We will show that $d_f = \mathsf{polylog}(N)$ in Section 4.4. With this, we can see that $\left\|\widetilde{\mathbf{e}} - \mathbf{e}^\top\mathbf{H}_{S,i}\mathbf{G}^{-1}(\mathbf{P})\right\| \leq \sqrt{m}\widetilde{\chi} + \sqrt{m}\lceil \log N \rceil \chi m^{O(d_f)} \leq \chi\lceil \log N \rceil \widetilde{\chi} m^{O(d_f)} = B$. Thus, as we chose tg to not in $[-\lfloor q/4 \rfloor - B, -\lfloor q/4 \rfloor + B] \cup [\lfloor q/4 \rfloor - B, \lfloor q/4 \rfloor + B]$, we have that $\lceil \mathsf{tg} \rfloor_2 = \lceil \mathsf{tg}' \rfloor_2$. Thus, $\mu' = h \oplus \lceil \mathsf{tg}' \rfloor_2 = \mu \oplus \lceil \mathsf{tg} \rfloor_2 \oplus \lceil \mathsf{tg}' \rfloor_2 = \mu$.

Remark 4.7 (Random public-key ciphertexts)**.** Note that pkEnc in Construction 4.6 outputs uniformly random strings of appropriate length. This will yield a transparent setup in applications of PPS schemes like distributed broadcast encryption [GY26c].

Remark 4.8 (Larger message space)**.** As already observed in [GY26c], we can use standard hybrid encryption techniques to encrypt longer messages.

Remark 4.9 (Encoding $\mathbb{Z}_q$ elements)**.** As q is prime, C (Figure 1) outputs each $x \in \mathbb{Z}_q$ as the ℓ_q-bit binary representation of $x + qr$ where $\ell_q := \lceil \log q \rceil + \lambda$ and $r \leftarrow [0, \lfloor 2^{\ell_q}/q \rfloor - 1]$ is sampled by skEnc and hardwired in C. Decode reduces it modulo q. If x is uniformly random in $\mathbb{Z}_q$, its encoding is within statistical distance $q/2^{\ell_q} \leq 2^{-\lambda}$ of a uniformly random string in $\{0,1\}^{\ell_q}$.

4.4 Full Compactness Analysis

We begin with a few observations that will useful in the analysis later on.

Claim 4.10. For any $S \subseteq [N]$, the set membership circuit f_S considered in Step 1 of KGen and Step 3 of Decode in Construction 4.6 has a boolean circuit representation with size $s_f = O(|S| \cdot \lceil \log N \rceil)$ and depth $d_f = O(\log N)$.

Proof. We write $f_S(\mathbf{x}) = \bigvee_{s \in S} \bigwedge_{j \in [\lceil \log N \rceil]} (x_j \oplus s_j \oplus 1)$ where s_j is the j-th bit of $\mathsf{encvec}(s)$. As s_j is fixed, each $x_j \oplus s_j \oplus 1$ is either x_j or its negation which requires one NAND gate. The AND of $\lceil \log N \rceil$ bits can be computed by a binary tree of depth $\lceil \log \lceil \log N \rceil \rceil$ and the OR of $|S|$ bits can be computed by a binary tree of depth $\lceil \log |S| \rceil \leq \lceil \log N \rceil$. Each AND and OR gate requires at most 3 NAND gates arranged in 2 levels. So, $d_f \leq 1 + 2\lceil \log \lceil \log N \rceil \rceil + 2\lceil \log N \rceil = O(\log N)$ and $s_f \leq 3|S|\lceil \log N \rceil + 3|S| = O(|S| \cdot \lceil \log N \rceil)$. Note that the homomorphic computation machinery allows arbitrary fan-out as $\|\mathbf{H}_{f,\mathbf{x}}\|$ only depends on the depth. As $|S| \leq N$, the bound on d_f holds for every $S \subseteq [N]$. □

Claim 4.11. For the circuit C from Figure 1, there exists a boolean circuit representation with size $s_C = \mathsf{poly}(\lambda, n, m, \lceil \log N \rceil, \lceil \log q \rceil)$. Moreover, s_C does not depend on S, i, or μ.

Proof. The input of C has $nm\lceil \log q \rceil$ bits. The hardwired values consist of $O(nm\lceil \log N \rceil + n\lambda)$ elements of $\mathbb{Z}_q$, the index i, the message μ, and the randomness of the encoding of Remark 4.9, which have $\mathsf{poly}(\lambda, n, m, \lceil \log N \rceil, \lceil \log q \rceil)$ bits in total. Computing $\mathbf{c}^\top$ requires $O(nm\lceil \log N \rceil)$ arithmetic operations in $\mathbb{Z}_q$. As $\mathbf{G}^{-1}(\mathbf{P}) \in \{0,1\}^{m \times \lambda}$ is the bit decomposition of the hardwired $\mathbf{P}$, it requires no gates. So, computing $\mathbf{B}\mathbf{G}^{-1}(\mathbf{P})$ requires $O(nm\lambda)$ additions in $\mathbb{Z}_q$ and computing $\mathbf{d}^\top$ from it requires $O(n\lambda)$ additional arithmetic operations in $\mathbb{Z}_q$. Each arithmetic operation in $\mathbb{Z}_q$, including the reduction modulo q, requires $O(\lceil \log q \rceil^2)$ NAND gates. The encoding of Remark 4.9 adds $O(m\lceil \log N \rceil + n\lambda + \lambda)$ additions and multiplications of ℓ_q-bit integers, each of which requires $O(\ell_q^2)$ NAND gates. Thus,

$$s_C = O\left(nm(\lceil \log N \rceil + \lambda)\lceil \log q \rceil^2 + (m\lceil \log N \rceil + n\lambda + \lambda)\ell_q^2\right) = \mathsf{poly}(\lambda, n, m, \lceil \log N \rceil, \lceil \log q \rceil).$$

The input of C is the digest $\mathbf{A}_S \in \mathbb{Z}_q^{n \times m}$ whose size does not depend on S. Moreover, i and μ only change the hardwired values of C and not its topology. So, s_C does not depend on S, i, or μ. In particular, N affects s_C only through $\lceil \log N \rceil$ and the LWE parameters n, m, q. □

Claim 4.12 ([BLSV18, WW24])**.** For the s_C from Claim 4.11, the length of the garbled circuit $\widetilde{C}$ is $L_{\widetilde{C}} = \mathsf{poly}(\lambda, n, m, \lceil \log N \rceil, \lceil \log q \rceil)$

Proof. The output of C has length $\lambda + (m\lceil \log N \rceil + n\lambda + \lambda)\ell_q$ where $\ell_q = \lceil \log q \rceil + \lambda$ (Remark 4.9). Then, the size of point-and-permute garbling used in [WW24] is $\lambda + 4(\lambda+1)s_C + \lambda + (m\lceil \log N \rceil + n\lambda + \lambda)\ell_q = \mathsf{poly}(\lambda, n, m, \lceil \log N \rceil, \lceil \log q \rceil)$. □

From Claim 4.10, Claim 4.11, Claim 4.12, we can see that

$$\begin{aligned}
|\mathsf{crs}| &= nm\lceil \log N \rceil \lceil \log q \rceil = \mathsf{poly}(\lambda, n, m, \lceil \log N \rceil, \lceil \log q \rceil) \\
|\mathsf{msk}| &= O(\lambda^2 L_{\widetilde{C}}) + nm\lceil \log q \rceil(2\lambda + 1) = \mathsf{poly}(\lambda, n, m, \lceil \log N \rceil, \lceil \log q \rceil) \\
|\mathsf{ct}| &= \lambda + L_{\widetilde{C}} = \mathsf{poly}(\lambda, n, m, \lceil \log N \rceil, \lceil \log q \rceil) \\
|\mathsf{sk}_S| &= O(\lambda^2 L_{\widetilde{C}}) + nm\lceil \log q \rceil(\lambda + 1) = \mathsf{poly}(\lambda, n, m, \lceil \log N \rceil, \lceil \log q \rceil)
\end{aligned}$$

Here we also used Theorem 4.3 and Theorem 4.5. By the LWE parameters we considered, each parameter size is bounded by $\mathsf{poly}(\lambda, \log N)$.

Moreover, we also point out that unlike [GY26c], none of our algorithms have running times growing with N (beyond the set description $|S|$) and are all efficient.

4.5 Mode Indistinguishability Analysis

Theorem 4.13 (Mode Indistinguishability)**.** If SEPRF satisfies security property (Definition 4.2), Garb satisfies security and blindness properties (Definition 4.4), and LWE assumption holds (Assumption 3.4), then Construction 4.6 satisfies mode indistinguishability property (Definition 4.1).

Proof. We prove this theorem using the following hybrids for an admissible adversary $\mathcal{A}$. Let $Q = Q(\lambda)$ be a polynomial that upper bounds the number of oracle queries made by $\mathcal{A}$. We call $\mathsf{ct}_{i^*} = (t^*, T^*)$ the target ciphertext and the circuit C^* garbled inside it the target circuit. In every hybrid, each oracle query (i, μ_i) is answered using a fresh circuit C_i and a fresh real garbling of C_i. Only the target ciphertext and the PRF key in sk_S change across the hybrids.

$\mathbf{Hyb}_0^{\mathcal{A}}(1^\lambda)$. This is the mode indistinguishability experiment from Definition 4.1 with $b = 1$.

Setup Phase. $\mathcal{A}$ sends (N, i^*). Sample $\mathbf{A} \leftarrow \mathbb{Z}_q^{n \times (\lceil \log N \rceil \cdot m)}$ and set $\mathsf{crs} := \mathbf{A}$. Sample $\mathsf{prf.key} \leftarrow \mathsf{SEPRF.Setup}(1^\lambda, 1^{L_{\tilde{C}}})$ and $\mathsf{garb.gk} \leftarrow \mathsf{Garb.Setup}(1^\lambda, 1^{s_C}, 1^{n \cdot m \cdot \lceil \log q \rceil})$. Set $\mathsf{msk} := (\mathsf{prf.key}, \mathsf{garb.gk})$. Sample $\mu_{i^*} \leftarrow \{0,1\}^\lambda$. Sample $\mathbf{s}, \mathbf{e}, \mathbf{P}, \widetilde{\mathbf{e}}, \mathsf{tg}$ as in skEnc and construct the circuit C^* from Figure 1 for index i^* and message μ_{i^*}. Sample $\widetilde{C}^* \leftarrow \mathsf{Garb.CircGarb}(\mathsf{garb.gk}, C^*)$ and $t^* \leftarrow \{0,1\}^\lambda$. Compute $T^* = \mathsf{SEPRF.Eval}(\mathsf{prf.key}, t^*) \oplus \widetilde{C}^*$. Send crs and $\mathsf{ct}_{i^*} := (t^*, T^*)$ to $\mathcal{A}$.

Query Phase. Given (i, μ_i) such that $i \neq i^*$, construct the circuit C_i from Figure 1 for index i and message μ_i using fresh randomness. Sample $\widetilde{C}_i \leftarrow \mathsf{Garb.CircGarb}(\mathsf{garb.gk}, C_i)$ and $t_i \leftarrow \{0,1\}^\lambda$. Compute $T_i = \mathsf{SEPRF.Eval}(\mathsf{prf.key}, t_i) \oplus \widetilde{C}_i$. Send $\mathsf{ct}_i := (t_i, T_i)$ to $\mathcal{A}$.

Challenge Phase. $\mathcal{A}$ sends $S \subseteq [N]$ such that $i^* \in S$. Compute $\mathbf{A}_S = \mathsf{EvalF}(\mathbf{A}, f_S)$ and $\widetilde{\mathbf{A}}_S \leftarrow \mathsf{Garb.InpGarb}(\mathsf{garb.gk}, \mathbf{A}_S)$. Send $\mathsf{sk}_S := (\mathsf{prf.key}, \widetilde{\mathbf{A}}_S)$ to $\mathcal{A}$.

Guess Phase. $\mathcal{A}$ outputs its guess b'. Output b'.

$\mathbf{Hyb}_1^{\mathcal{A}}(1^\lambda)$. This is the same as $\mathbf{Hyb}_0^{\mathcal{A}}(1^\lambda)$ except that we sample $t^* \leftarrow \{0,1\}^\lambda$ at the beginning of the setup phase. During the query phase, if $t_i = t^*$ for any query (i, μ_i), the challenger aborts and outputs 0.

$\mathbf{Hyb}_2^{\mathcal{A}}(1^\lambda)$. This is the same as $\mathbf{Hyb}_1^{\mathcal{A}}(1^\lambda)$ except that we sample $\mathsf{eqKey} \leftarrow \mathsf{SEPRF.Sim}_1(1^\lambda, 1^{L_{\tilde{C}}}, t^*)$, $y^* \leftarrow \{0,1\}^{L_{\tilde{C}}}$, and set $\mathsf{prf.key} \leftarrow \mathsf{SEPRF.Sim}_2(\mathsf{eqKey}, y^*)$ during the setup phase. Note that $T^* = y^* \oplus \widetilde{C}^*$ in this hybrid by correctness of SEPRF.

$\mathbf{Hyb}_3^{\mathcal{A}}(1^\lambda)$. This is the same as $\mathbf{Hyb}_2^{\mathcal{A}}(1^\lambda)$ except that we sample T^* uniformly during the setup phase and defer the sampling of μ_{i^*}, the target circuit, and $\mathsf{prf.key}$ to the challenge phase. We write the full hybrid for clarity.

Setup Phase. $\mathcal{A}$ sends (N, i^*). Sample $t^* \leftarrow \{0,1\}^\lambda$. Sample $\mathbf{A} \leftarrow \mathbb{Z}_q^{n \times (\lceil \log N \rceil \cdot m)}$ and set $\mathsf{crs} := \mathbf{A}$. Sample $\mathsf{garb.gk} \leftarrow \mathsf{Garb.Setup}(1^\lambda, 1^{s_C}, 1^{n \cdot m \cdot \lceil \log q \rceil})$ and $\mathsf{eqKey} \leftarrow \mathsf{SEPRF.Sim}_1(1^\lambda, 1^{L_{\tilde{C}}}, t^*)$. Sample $T^* \leftarrow \{0,1\}^{L_{\tilde{C}}}$. Send crs and $\mathsf{ct}_{i^*} := (t^*, T^*)$ to $\mathcal{A}$.

Query Phase. Given (i, μ_i) such that $i \neq i^*$, construct the circuit C_i from Figure 1 for index i and message μ_i using fresh randomness. Sample $\widetilde{C}_i \leftarrow \mathsf{Garb.CircGarb}(\mathsf{garb.gk}, C_i)$ and $t_i \leftarrow \{0,1\}^\lambda$. If $t_i = t^*$, abort and output 0. Otherwise, compute $T_i = \mathsf{SEPRF.Eval}(\mathsf{eqKey}, t_i) \oplus \widetilde{C}_i$. Send $\mathsf{ct}_i := (t_i, T_i)$ to $\mathcal{A}$.

Challenge Phase. $\mathcal{A}$ sends $S \subseteq [N]$ such that $i^* \in S$. Compute $\mathbf{A}_S = \mathsf{EvalF}(\mathbf{A}, f_S)$ and $\widetilde{\mathbf{A}}_S \leftarrow \mathsf{Garb.InpGarb}(\mathsf{garb.gk}, \mathbf{A}_S)$. Sample $\mu_{i^*} \leftarrow \{0,1\}^\lambda$. Sample $\mathbf{s}, \mathbf{e}, \mathbf{P}, \widetilde{\mathbf{e}}, \mathsf{tg}$ as in skEnc and construct the circuit C^* from Figure 1 for index i^* and message μ_{i^*}. Sample $\widetilde{C}^* \leftarrow \mathsf{Garb.CircGarb}(\mathsf{garb.gk}, C^*)$. Compute $y^* = T^* \oplus \widetilde{C}^*$ and sample $\mathsf{prf.key} \leftarrow \mathsf{SEPRF.Sim}_2(\mathsf{eqKey}, y^*)$. Send $\mathsf{sk}_S := (\mathsf{prf.key}, \widetilde{\mathbf{A}}_S)$ to $\mathcal{A}$.

Guess Phase. $\mathcal{A}$ outputs its guess b'. Output b'.

$\mathbf{Hyb}_4^{\mathcal{A}}(1^\lambda)$. This is the same as $\mathbf{Hyb}_3^{\mathcal{A}}(1^\lambda)$ except that during the challenge phase, we compute $(\mathbf{c}^\top, \mathbf{P}, \mathbf{d}^\top, h) = C^*(\mathbf{A}_S)$ and sample $\widetilde{C}^* \leftarrow \mathsf{Garb.Sim}(\mathsf{garb.gk}, \mathbf{A}_S, (\mathbf{c}^\top, \mathbf{P}, \mathbf{d}^\top, h))$. Here, $\mathbf{c}^\top = \mathbf{s}^\top(\mathbf{A} - \mathsf{encvec}(i^*) \otimes \mathbf{G}) + \mathbf{e}^\top$, $\mathbf{d}^\top = \mathbf{s}^\top(\mathbf{A}_S \mathbf{G}^{-1}(\mathbf{P})) - \mathbf{s}^\top \mathbf{P} + \widetilde{\mathbf{e}}^\top + \mathsf{tg}$, and $h = \lceil \mathsf{tg} \rfloor_2 \oplus \mu_{i^*}$.

$\mathbf{Hyb}_5^{\mathcal{A}}(1^\lambda)$. This is the same as $\mathbf{Hyb}_4^{\mathcal{A}}(1^\lambda)$ except that during the challenge phase, we compute $\mathbf{H}_{S,i^*} = \mathsf{EvalFX}(\mathbf{A}, f_S, \mathsf{encvec}(i^*))$ and set $\mathbf{d}^\top = \mathbf{c}^\top \mathbf{H}_{S,i^*} \mathbf{G}^{-1}(\mathbf{P}) + \widetilde{\mathbf{e}}^\top + \mathsf{tg}$.

$\mathbf{Hyb}_6^{\mathcal{A}}(1^\lambda)$. This is the same as $\mathbf{Hyb}_5^{\mathcal{A}}(1^\lambda)$ except that we sample $\mathbf{c}^\top \leftarrow \mathbb{Z}_q^{1 \times (\lceil \log N \rceil \cdot m)}$ uniformly during the challenge phase. We write the full hybrid for clarity.

Setup Phase. $\mathcal{A}$ sends (N, i^*). Sample $t^* \leftarrow \{0,1\}^\lambda$. Sample $\mathbf{A} \leftarrow \mathbb{Z}_q^{n \times (\lceil \log N \rceil \cdot m)}$ and set $\mathsf{crs} := \mathbf{A}$. Sample $\mathsf{garb.gk} \leftarrow \mathsf{Garb.Setup}(1^\lambda, 1^{s_C}, 1^{n \cdot m \cdot \lceil \log q \rceil})$ and $\mathsf{eqKey} \leftarrow \mathsf{SEPRF.Sim}_1(1^\lambda, 1^{L_{\tilde{C}}}, t^*)$. Sample $T^* \leftarrow \{0,1\}^{L_{\tilde{C}}}$. Send crs and $\mathsf{ct}_{i^*} := (t^*, T^*)$ to $\mathcal{A}$.

Query Phase. Same as $\mathbf{Hyb}_3^{\mathcal{A}}(1^\lambda)$.

Challenge Phase. $\mathcal{A}$ sends $S \subseteq [N]$ such that $i^* \in S$. Compute $\mathbf{A}_S = \mathsf{EvalF}(\mathbf{A}, f_S)$, $\widetilde{\mathbf{A}}_S \leftarrow \mathsf{Garb.InpGarb}(\mathsf{garb.gk}, \mathbf{A}_S)$, and $\mathbf{H}_{S,i^*} = \mathsf{EvalFX}(\mathbf{A}, f_S, \mathsf{encvec}(i^*))$. Sample $\mu_{i^*} \leftarrow \{0,1\}^\lambda$. Sample $\mathbf{c}^\top \leftarrow \mathbb{Z}_q^{1 \times (\lceil \log N \rceil \cdot m)}$. Sample $\mathbf{P}, \widetilde{\mathbf{e}}, \mathsf{tg}$ as in skEnc. Compute $\mathbf{d}^\top = \mathbf{c}^\top \mathbf{H}_{S,i^*} \mathbf{G}^{-1}(\mathbf{P}) + \widetilde{\mathbf{e}}^\top + \mathsf{tg}$ and $h = \lceil \mathsf{tg} \rfloor_2 \oplus \mu_{i^*}$. Sample $\widetilde{C}^* \leftarrow \mathsf{Garb.Sim}(\mathsf{garb.gk}, \mathbf{A}_S, (\mathbf{c}^\top, \mathbf{P}, \mathbf{d}^\top, h))$. Compute $y^* = T^* \oplus \widetilde{C}^*$ and sample $\mathsf{prf.key} \leftarrow \mathsf{SEPRF.Sim}_2(\mathsf{eqKey}, y^*)$. Send $\mathsf{sk}_S := (\mathsf{prf.key}, \widetilde{\mathbf{A}}_S)$ to $\mathcal{A}$.

Guess Phase. $\mathcal{A}$ outputs its guess b'. Output b'.

$\mathbf{Hyb}_7^{\mathcal{A}}(1^\lambda)$. This is the same as $\mathbf{Hyb}_6^{\mathcal{A}}(1^\lambda)$ except that we sample $\mathsf{tg} \leftarrow \mathbb{Z}_q^\lambda$ uniformly during the challenge phase.

$\mathbf{Hyb}_8^{\mathcal{A}}(1^\lambda)$. This is the same as $\mathbf{Hyb}_7^{\mathcal{A}}(1^\lambda)$ except that we sample $\mathbf{d}^\top \leftarrow \mathbb{Z}_q^{1 \times \lambda}$ and $h \leftarrow \{0,1\}^\lambda$ uniformly during the challenge phase. Note that we no longer need to sample $\mu_{i^*}, \widetilde{\mathbf{e}}, \mathsf{tg}$.

$\mathbf{Hyb}_9^{\mathcal{A}}(1^\lambda)$. This is the same as $\mathbf{Hyb}_8^{\mathcal{A}}(1^\lambda)$ except that we sample $\widetilde{C}^* \leftarrow \{0,1\}^{L_{\tilde{C}}}$ uniformly during the challenge phase. We write the full hybrid for clarity.

Setup Phase. Same as $\mathbf{Hyb}_6^{\mathcal{A}}(1^\lambda)$.

Query Phase. Same as $\mathbf{Hyb}_3^{\mathcal{A}}(1^\lambda)$.

Challenge Phase. $\mathcal{A}$ sends $S \subseteq [N]$ such that $i^* \in S$. Compute $\mathbf{A}_S = \mathsf{EvalF}(\mathbf{A}, f_S)$ and $\widetilde{\mathbf{A}}_S \leftarrow \mathsf{Garb.InpGarb}(\mathsf{garb.gk}, \mathbf{A}_S)$. Sample $\widetilde{C}^* \leftarrow \{0,1\}^{L_{\tilde{C}}}$. Compute $y^* = T^* \oplus \widetilde{C}^*$ and sample $\mathsf{prf.key} \leftarrow \mathsf{SEPRF.Sim}_2(\mathsf{eqKey}, y^*)$. Send $\mathsf{sk}_S := (\mathsf{prf.key}, \widetilde{\mathbf{A}}_S)$ to $\mathcal{A}$.

Guess Phase. $\mathcal{A}$ outputs its guess b'. Output b'.

$\mathbf{Hyb}_{10}^{\mathcal{A}}(1^\lambda)$. This is the same as $\mathbf{Hyb}_9^{\mathcal{A}}(1^\lambda)$ except that we sample $y^* \leftarrow \{0,1\}^{L_{\tilde{C}}}$ and $\mathsf{prf.key} \leftarrow \mathsf{SEPRF.Sim}_2(\mathsf{eqKey}, y^*)$ during the setup phase. During the query phase, we compute $T_i = \mathsf{SEPRF.Eval}(\mathsf{prf.key}, t_i) \oplus \widetilde{C}_i$. During the challenge phase, we only compute $\widetilde{\mathbf{A}}_S$ and send $\mathsf{sk}_S := (\mathsf{prf.key}, \widetilde{\mathbf{A}}_S)$.

$\mathbf{Hyb}_{11}^{\mathcal{A}}(1^\lambda)$. This is the same as $\mathbf{Hyb}_{10}^{\mathcal{A}}(1^\lambda)$ except that we sample $\mathsf{prf.key} \leftarrow \mathsf{SEPRF.Setup}(1^\lambda, 1^{L_{\tilde{C}}})$ during the setup phase. Note that we no longer need to sample eqKey and y^*.

$\mathbf{Hyb}_{12}^{\mathcal{A}}(1^\lambda)$. This is the same as $\mathbf{Hyb}_{11}^{\mathcal{A}}(1^\lambda)$ except that the challenger no longer aborts when $t_i = t^*$ for some query. We write the full hybrid for clarity. Note that this is the mode indistinguishability experiment from Definition 4.1 with $b = 0$.

Setup Phase. $\mathcal{A}$ sends (N, i^*). Sample $\mathbf{A} \leftarrow \mathbb{Z}_q^{n \times (\lceil \log N \rceil \cdot m)}$ and set $\mathsf{crs} := \mathbf{A}$. Sample $\mathsf{prf.key} \leftarrow \mathsf{SEPRF.Setup}(1^\lambda, 1^{L_{\tilde{C}}})$ and $\mathsf{garb.gk} \leftarrow \mathsf{Garb.Setup}(1^\lambda, 1^{s_C}, 1^{n \cdot m \cdot \lceil \log q \rceil})$. Set $\mathsf{msk} := (\mathsf{prf.key}, \mathsf{garb.gk})$. Sample $t^* \leftarrow \{0,1\}^\lambda$ and $T^* \leftarrow \{0,1\}^{L_{\tilde{C}}}$, i.e., $\mathsf{ct}_{i^*} := (t^*, T^*) \leftarrow \mathsf{pkEnc}(i^*)$. Send crs and ct_{i^*} to $\mathcal{A}$.

Query Phase. Given (i, μ_i) such that $i \neq i^*$, construct the circuit C_i from Figure 1 for index i and message μ_i using fresh randomness. Sample $\widetilde{C}_i \leftarrow \mathsf{Garb.CircGarb}(\mathsf{garb.gk}, C_i)$ and $t_i \leftarrow \{0,1\}^\lambda$. Compute $T_i = \mathsf{SEPRF.Eval}(\mathsf{prf.key}, t_i) \oplus \widetilde{C}_i$. Send $\mathsf{ct}_i := (t_i, T_i)$ to $\mathcal{A}$.

Challenge Phase. $\mathcal{A}$ sends $S \subseteq [N]$ such that $i^* \in S$. Compute $\mathbf{A}_S = \mathsf{EvalF}(\mathbf{A}, f_S)$ and $\widetilde{\mathbf{A}}_S \leftarrow \mathsf{Garb.InpGarb}(\mathsf{garb.gk}, \mathbf{A}_S)$. Send $\mathsf{sk}_S := (\mathsf{prf.key}, \widetilde{\mathbf{A}}_S)$ to $\mathcal{A}$.

Guess Phase. $\mathcal{A}$ outputs its guess b'. Output b'.

Lemma 4.14. For any PPT adversary $\mathcal{A}$, $|\Pr[1 \leftarrow \mathbf{Hyb}_0^{\mathcal{A}}(1^\lambda)] - \Pr[1 \leftarrow \mathbf{Hyb}_1^{\mathcal{A}}(1^\lambda)]| \leq Q \cdot 2^{-\lambda}$.

Proof. Sampling t^* at the beginning of the setup phase does not change the view of $\mathcal{A}$. Moreover, each t_i is sampled uniformly at random and independently of t^*. So, by a union bound, the probability that the challenger aborts in $\mathbf{Hyb}_1^{\mathcal{A}}(1^\lambda)$ is at most $Q \cdot 2^{-\lambda}$. Conditioned on the challenger not aborting, both hybrids are identical. Thus, the lemma holds. □

Lemma 4.15. If SEPRF satisfies security property (Definition 4.2), then for any PPT admissible adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for any $\lambda \in \mathbb{N}$, $|\Pr[1 \leftarrow \mathbf{Hyb}_1^{\mathcal{A}}(1^\lambda)] - \Pr[1 \leftarrow \mathbf{Hyb}_2^{\mathcal{A}}(1^\lambda)]| \leq \mathsf{negl}(\lambda)$.

Proof. Assume for the sake of contradiction that there exists an adversary $\mathcal{A}$ such that $|\Pr[1 \leftarrow \mathbf{Hyb}_1^{\mathcal{A}}(1^\lambda)] - \Pr[1 \leftarrow \mathbf{Hyb}_2^{\mathcal{A}}(1^\lambda)]| = \epsilon(\lambda)$ for a non-negligible function $\epsilon(\cdot)$. We will construct a reduction adversary $\mathcal{B}$ that breaks the security property of SEPRF with non-negligible advantage. Description of $\mathcal{B}$ is as follows.

$\mathcal{B}$ receives (N, i^*) from $\mathcal{A}$. It samples $t^* \leftarrow \{0,1\}^\lambda$, sends $(1^{L_{\bar{C}}}, t^*)$ to the SEPRF challenger, and receives key. $\mathcal{B}$ samples $\mathbf{A}$ and $\mathsf{garb.gk}$ on its own and computes the target ciphertext (t^*, T^*) similarly to $\mathbf{Hyb}_1^{\mathcal{A}}(1^\lambda)$ using key in place of $\mathsf{prf.key}$. It sends $\mathsf{crs} := \mathbf{A}$ and (t^*, T^*) to $\mathcal{A}$. $\mathcal{B}$ answers each query (i, μ_i) similarly to $\mathbf{Hyb}_1^{\mathcal{A}}(1^\lambda)$ using key and $\mathsf{garb.gk}$. If $t_i = t^*$ for any query, $\mathcal{B}$ aborts and outputs 0. When $\mathcal{A}$ sends S, $\mathcal{B}$ computes $\mathbf{A}_S = \mathsf{EvalF}(\mathbf{A}, f_S)$, $\widetilde{\mathbf{A}}_S \leftarrow \mathsf{Garb.InpGarb}(\mathsf{garb.gk}, \mathbf{A}_S)$, and sends $\mathsf{sk}_S := (\mathsf{key}, \widetilde{\mathbf{A}}_S)$ to $\mathcal{A}$. $\mathcal{B}$ outputs the guess b' of $\mathcal{A}$ as its own.

If $\mathsf{key} \leftarrow \mathsf{SEPRF.Setup}(1^\lambda, 1^{L_{\bar{C}}})$, then $\mathcal{B}$ behaves like $\mathbf{Hyb}_1^{\mathcal{A}}(1^\lambda)$. Otherwise, $\mathsf{key} \leftarrow \mathsf{SEPRF.Sim}_2(\mathsf{eqKey}, y^*)$ for $\mathsf{eqKey} \leftarrow \mathsf{SEPRF.Sim}_1(1^\lambda, 1^{L_{\bar{C}}}, t^*)$ and a uniformly random y^*, and $\mathcal{B}$ behaves like $\mathbf{Hyb}_2^{\mathcal{A}}(1^\lambda)$. Thus, $\mathcal{B}$ breaks the security property of SEPRF with advantage $\epsilon(\lambda)$. Thus, the lemma holds. □

Lemma 4.16. $\Pr[1 \leftarrow \mathbf{Hyb}_2^{\mathcal{A}}(1^\lambda)] = \Pr[1 \leftarrow \mathbf{Hyb}_3^{\mathcal{A}}(1^\lambda)]$.

Proof. In $\mathbf{Hyb}_2^{\mathcal{A}}(1^\lambda)$, y^* is uniformly random and is only used to compute $\mathsf{prf.key} \leftarrow \mathsf{SEPRF.Sim}_2(\mathsf{eqKey}, y^*)$. By correctness of SEPRF, $T^* = y^* \oplus \widetilde{C}^*$. So, T^* is uniformly random and independent of $\widetilde{C}^*$ and $y^* = T^* \oplus \widetilde{C}^*$. Moreover, for each query, $t_i \neq t^*$ unless the challenger aborts. So, $\mathsf{SEPRF.Eval}(\mathsf{prf.key}, t_i) = \mathsf{SEPRF.Eval}(\mathsf{eqKey}, t_i)$ by correctness of SEPRF. Thus, $\mathsf{prf.key}$ is not needed until the challenge phase. Finally, μ_{i^*} and the target circuit C^* are not used before the challenge phase and C^* does not depend on S. So, we can sample them during the challenge phase. Hence, the output distributions of $\mathbf{Hyb}_2^{\mathcal{A}}(1^\lambda)$ and $\mathbf{Hyb}_3^{\mathcal{A}}(1^\lambda)$ are identical and the lemma holds. □

Lemma 4.17. If Garb satisfies security property (Definition 4.4), then for any PPT admissible adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for any $\lambda \in \mathbb{N}$, $|\Pr[1 \leftarrow \mathbf{Hyb}_3^{\mathcal{A}}(1^\lambda)] - \Pr[1 \leftarrow \mathbf{Hyb}_4^{\mathcal{A}}(1^\lambda)]| \leq \mathsf{negl}(\lambda)$.

Proof. Assume for the sake of contradiction that there exists an adversary $\mathcal{A}$ such that $|\Pr[1 \leftarrow \mathbf{Hyb}_3^{\mathcal{A}}(1^\lambda)] - \Pr[1 \leftarrow \mathbf{Hyb}_4^{\mathcal{A}}(1^\lambda)]| = \epsilon(\lambda)$ for a non-negligible function $\epsilon(\cdot)$. We will construct a reduction adversary $\mathcal{B}$ that breaks the security property of Garb with non-negligible advantage. Description of $\mathcal{B}$ is as follows.

$\mathcal{B}$ sends $(1^{s_C}, 1^{n \cdot m \cdot \lceil \log q \rceil})$ to the Garb challenger and gets access to the oracle $\mathsf{Garb.CircGarb}(\mathsf{garb.gk}, \cdot)$. $\mathcal{B}$ receives (N, i^*) from $\mathcal{A}$. It samples $t^*, \mathbf{A}, \mathsf{eqKey}, T^*$ similarly to $\mathbf{Hyb}_3^{\mathcal{A}}(1^\lambda)$ and sends $\mathsf{crs} := \mathbf{A}$ and $\mathsf{ct}_{i^*} := (t^*, T^*)$ to $\mathcal{A}$. For each query (i, μ_i), $\mathcal{B}$ constructs the circuit C_i and queries its oracle with C_i to receive $\widetilde{C}_i$. It then samples t_i and computes T_i similarly to $\mathbf{Hyb}_3^{\mathcal{A}}(1^\lambda)$, including the abort. When $\mathcal{A}$ sends S, $\mathcal{B}$ computes $\mathbf{A}_S = \mathsf{EvalF}(\mathbf{A}, f_S)$ and sends $\mathbf{A}_S$ to the Garb challenger as its challenge input. It receives $\widetilde{\mathbf{A}}_S$. $\mathcal{B}$ then samples $\mu_{i^*} \leftarrow \{0,1\}^\lambda$ and $\mathbf{s}, \mathbf{e}, \mathbf{P}, \widetilde{\mathbf{e}}, \mathsf{tg}$ as in skEnc, constructs the target circuit C^*, and sends C^* to the Garb challenger as its challenge circuit. It receives $\widetilde{C}^*$. $\mathcal{B}$ computes $y^* = T^* \oplus \widetilde{C}^*$, samples $\mathsf{prf.key} \leftarrow \mathsf{SEPRF.Sim}_2(\mathsf{eqKey}, y^*)$, and sends $\mathsf{sk}_S := (\mathsf{prf.key}, \widetilde{\mathbf{A}}_S)$ to $\mathcal{A}$. $\mathcal{B}$ outputs the guess b' of $\mathcal{A}$ as its own.

Note that $\mathcal{B}$ makes exactly one challenge input garbling query followed by one challenge circuit garbling query. So, $\mathcal{B}$ is an admissible adversary against Garb. If $\widetilde{C}^* \leftarrow \mathsf{Garb.CircGarb}(\mathsf{garb.gk}, C^*)$, then $\mathcal{B}$ behaves like $\mathbf{Hyb}_3^{\mathcal{A}}(1^\lambda)$. Otherwise, $\widetilde{C}^* \leftarrow \mathsf{Garb.Sim}(\mathsf{garb.gk}, \mathbf{A}_S, C^*(\mathbf{A}_S))$ and $\mathcal{B}$ behaves like $\mathbf{Hyb}_4^{\mathcal{A}}(1^\lambda)$. Thus, $\mathcal{B}$ breaks the security property of Garb with advantage $\epsilon(\lambda)$. Thus, the lemma holds. □

Lemma 4.18. For any PPT admissible adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for any $\lambda \in \mathbb{N}$, $|\Pr[1 \leftarrow \mathbf{Hyb}_4^{\mathcal{A}}(1^\lambda)] - \Pr[1 \leftarrow \mathbf{Hyb}_5^{\mathcal{A}}(1^\lambda)]| \leq \mathsf{negl}(\lambda)$.

Proof. By the relations of matrix homomorphic computation machinery and as $f_S(i^*) = 1$ by admissibility of $\mathcal{A}$,

$$\begin{aligned}\mathbf{c}^\top \mathbf{H}_{S,i^*}\mathbf{G}^{-1}(\mathbf{P}) &= \mathbf{s}^\top(\mathbf{A}_S - \mathbf{G})\mathbf{G}^{-1}(\mathbf{P}) + \mathbf{e}^\top \mathbf{H}_{S,i^*}\mathbf{G}^{-1}(\mathbf{P}) \\ &= \mathbf{s}^\top(\mathbf{A}_S\mathbf{G}^{-1}(\mathbf{P})) - \mathbf{s}^\top\mathbf{P} + \mathbf{e}^\top \mathbf{H}_{S,i^*}\mathbf{G}^{-1}(\mathbf{P})\end{aligned}$$

So, the only difference between these hybrids is that $\mathbf{d}^\top$ in $\mathbf{Hyb}_5^{\mathcal{A}}(1^\lambda)$ is equal to $\mathbf{d}^\top$ in $\mathbf{Hyb}_4^{\mathcal{A}}(1^\lambda)$ shifted by $\mathbf{e}^\top \mathbf{H}_{S,i^*}\mathbf{G}^{-1}(\mathbf{P})$. As in the correctness analysis, $\left\|\mathbf{e}^\top \mathbf{H}_{S,i^*}\mathbf{G}^{-1}(\mathbf{P})\right\| \leq \sqrt{m}\lceil \log N \rceil \chi m^{O(d_f)}$ with overwhelming probability. Each entry of $\widetilde{\mathbf{e}}$ is sampled uniformly from $[-\sqrt{m}\widetilde{\chi}, \sqrt{m}\widetilde{\chi}]$ independently of all other values and $\widetilde{\chi} > \lambda^{\omega(1)} \cdot \chi \cdot \lceil \log N \rceil \cdot m^{\mathsf{polylog}(N)}$. Thus, by the smudging lemma (Lemma 3.3) applied to each of the λ entries, $\widetilde{\mathbf{e}}^\top$ and $\widetilde{\mathbf{e}}^\top + \mathbf{e}^\top \mathbf{H}_{S,i^*}\mathbf{G}^{-1}(\mathbf{P})$ are statistically indistinguishable even conditioned on all other values sampled in the hybrids. Hence, the lemma holds. □

Lemma 4.19. If LWE assumption holds (Assumption 3.4), then for any PPT admissible adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for any $\lambda \in \mathbb{N}$, $|\Pr[1 \leftarrow \mathbf{Hyb}_5^{\mathcal{A}}(1^\lambda)] - \Pr[1 \leftarrow \mathbf{Hyb}_6^{\mathcal{A}}(1^\lambda)]| \leq \mathsf{negl}(\lambda)$.

Proof. Note that in $\mathbf{Hyb}_5^{\mathcal{A}}(1^\lambda)$, $\mathbf{s}$ and $\mathbf{e}$ are only used to compute $\mathbf{c}^\top = \mathbf{s}^\top(\mathbf{A} - \mathsf{encvec}(i^*) \otimes \mathbf{G}) + \mathbf{e}^\top$. In particular, $\mathbf{d}^\top$ no longer depends on $\mathbf{s}$. Assume for the sake of contradiction that there exists an adversary $\mathcal{A}$ such that $|\Pr[1 \leftarrow \mathbf{Hyb}_5^{\mathcal{A}}(1^\lambda)] - \Pr[1 \leftarrow \mathbf{Hyb}_6^{\mathcal{A}}(1^\lambda)]| = \epsilon(\lambda)$ for a non-negligible function $\epsilon(\cdot)$. We will construct a reduction adversary $\mathcal{B}$ that breaks the LWE assumption with non-negligible advantage. Description of $\mathcal{B}$ is as follows.

$\mathcal{B}$ receives (N, i^*) from $\mathcal{A}$ and an LWE challenge $(\mathbf{M}, \mathbf{w}^\top)$ where $\mathbf{M} \in \mathbb{Z}_q^{n \times (\lceil \log N \rceil \cdot m)}$ and $\mathbf{w}^\top \in \mathbb{Z}_q^{1 \times (\lceil \log N \rceil \cdot m)}$. $\mathcal{B}$ sets $\mathbf{A} := \mathbf{M} + \mathsf{encvec}(i^*) \otimes \mathbf{G}$ which is uniformly random as $\mathbf{M}$ is. $\mathcal{B}$ samples $t^*, \mathsf{garb.gk}, \mathsf{eqKey}, T^*$ similarly to $\mathbf{Hyb}_5^{\mathcal{A}}(1^\lambda)$ and sends $\mathsf{crs} := \mathbf{A}$ and $\mathsf{ct}_{i^*} := (t^*, T^*)$ to $\mathcal{A}$. $\mathcal{B}$ answers each query (i, μ_i) similarly to $\mathbf{Hyb}_5^{\mathcal{A}}(1^\lambda)$, which only requires $\mathbf{A}$, $\mathsf{garb.gk}$, and eqKey. When $\mathcal{A}$ sends S, $\mathcal{B}$ computes $\mathbf{A}_S$, $\widetilde{\mathbf{A}}_S$, and $\mathbf{H}_{S,i^*}$. It samples $\mu_{i^*} \leftarrow \{0,1\}^\lambda$ and $\mathbf{P}, \widetilde{\mathbf{e}}, \mathsf{tg}$ as in skEnc. It sets $\mathbf{c}^\top := \mathbf{w}^\top$ and computes $\mathbf{d}^\top = \mathbf{c}^\top \mathbf{H}_{S,i^*}\mathbf{G}^{-1}(\mathbf{P}) + \widetilde{\mathbf{e}}^\top + \mathsf{tg}$ and $h = \lceil \mathsf{tg} \rfloor_2 \oplus \mu_{i^*}$. $\mathcal{B}$ samples $\widetilde{C}^* \leftarrow \mathsf{Garb.Sim}(\mathsf{garb.gk}, \mathbf{A}_S, (\mathbf{c}^\top, \mathbf{P}, \mathbf{d}^\top, h))$, computes $y^* = T^* \oplus \widetilde{C}^*$, samples $\mathsf{prf.key} \leftarrow \mathsf{SEPRF.Sim}_2(\mathsf{eqKey}, y^*)$, and sends $\mathsf{sk}_S := (\mathsf{prf.key}, \widetilde{\mathbf{A}}_S)$ to $\mathcal{A}$. $\mathcal{B}$ outputs the guess b' of $\mathcal{A}$ as its own.

If $\mathbf{w}^\top = \mathbf{s}^\top\mathbf{M} + \mathbf{e}^\top$, then $\mathbf{c}^\top = \mathbf{s}^\top(\mathbf{A} - \mathsf{encvec}(i^*) \otimes \mathbf{G}) + \mathbf{e}^\top$ and $\mathcal{B}$ behaves like $\mathbf{Hyb}_5^{\mathcal{A}}(1^\lambda)$. Otherwise, $\mathbf{w}^\top$ is uniformly random and $\mathcal{B}$ behaves like $\mathbf{Hyb}_6^{\mathcal{A}}(1^\lambda)$. Thus, $\mathcal{B}$ breaks the LWE assumption with advantage $\epsilon(\lambda)$. Thus, the lemma holds. □

Lemma 4.20. For any PPT admissible adversary $\mathcal{A}$, $|\Pr[1 \leftarrow \mathbf{Hyb}_6^{\mathcal{A}}(1^\lambda)] - \Pr[1 \leftarrow \mathbf{Hyb}_7^{\mathcal{A}}(1^\lambda)]| \leq \lambda(4B+2)/q$.

Proof. The only difference between these hybrids is the distribution of tg. In $\mathbf{Hyb}_6^{\mathcal{A}}(1^\lambda)$, each entry of tg is sampled uniformly from $\mathbb{Z}_q$ excluding $[-\lfloor q/4 \rfloor - B, -\lfloor q/4 \rfloor + B] \cup [\lfloor q/4 \rfloor - B, \lfloor q/4 \rfloor + B]$ which contains $4B+2$ elements. In $\mathbf{Hyb}_7^{\mathcal{A}}(1^\lambda)$, each entry is sampled uniformly from $\mathbb{Z}_q$. The statistical distance between these two distributions for each entry is at most $(4B+2)/q$. As tg has λ entries sampled independently of all other values, the statistical distance between the two hybrids is at most $\lambda(4B+2)/q$. This is negligible as $q > \lambda^{\omega(1)} \cdot \lambda \cdot (4B+2)$. Thus, the lemma holds. □

Lemma 4.21. $\Pr[1 \leftarrow \mathbf{Hyb}_7^{\mathcal{A}}(1^\lambda)] = \Pr[1 \leftarrow \mathbf{Hyb}_8^{\mathcal{A}}(1^\lambda)]$.

Proof. In $\mathbf{Hyb}_7^{\mathcal{A}}(1^\lambda)$, μ_{i^*} is uniformly random and is only used to compute $h = \lceil \mathsf{tg} \rfloor_2 \oplus \mu_{i^*}$. So, h is uniformly random in $\{0,1\}^\lambda$ and independent of tg and all other values. Moreover, tg is uniformly random in $\mathbb{Z}_q^\lambda$ and is now only used to compute $\mathbf{d}^\top = \mathbf{c}^\top \mathbf{H}_{S,i^*} \mathbf{G}^{-1}(\mathbf{P}) + \widetilde{\mathbf{e}}^\top + \mathsf{tg}$. So, $\mathbf{d}^\top$ is uniformly random in $\mathbb{Z}_q^{1\times\lambda}$ and independent of $(\mathbf{c}^\top, \mathbf{P}, h)$ and all other values. Hence, the output distributions of $\mathbf{Hyb}_7^{\mathcal{A}}(1^\lambda)$ and $\mathbf{Hyb}_8^{\mathcal{A}}(1^\lambda)$ are identical and the lemma holds. □

Lemma 4.22. If Garb satisfies blindness property (Definition 4.4), then for any PPT admissible adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for any $\lambda \in \mathbb{N}$, $|\Pr[1 \leftarrow \mathbf{Hyb}_8^{\mathcal{A}}(1^\lambda)] - \Pr[1 \leftarrow \mathbf{Hyb}_9^{\mathcal{A}}(1^\lambda)]| \le \mathsf{negl}(\lambda)$.

Proof. In $\mathbf{Hyb}_8^{\mathcal{A}}(1^\lambda)$, $\mathbf{c}^\top, \mathbf{P}, \mathbf{d}^\top$, and h are uniformly random and independent of all other values. The circuit C^* outputs $(\mathbf{c}^\top, \mathbf{P}, \mathbf{d}^\top, h)$ as a binary string using the encoding of Remark 4.9. This string consists of the encodings of $k := \lceil \log N \rceil m + n\lambda + \lambda$ independent and uniformly random elements of $\mathbb{Z}_q$ followed by h. So, by Remark 4.9, the binary string corresponding to $(\mathbf{c}^\top, \mathbf{P}, \mathbf{d}^\top, h)$ is within statistical distance $k \cdot 2^{-\lambda}$ of a uniformly random string of the output length of C^*. This is negligible as $n, m, \lceil \log N \rceil = \mathsf{poly}(\lambda)$. By blindness of Garb, $\mathsf{Garb.Sim}(\mathsf{garb.gk}, \mathbf{A}_S, \cdot)$ maps a uniformly random string of this length to a uniformly random string in $\{0,1\}^{L_{\bar{C}}}$. As $\mathsf{Garb.Sim}$ uses fresh randomness, applying it does not increase the statistical distance. Thus, $\widetilde{C}^*$ in $\mathbf{Hyb}_8^{\mathcal{A}}(1^\lambda)$ is statistically close to a uniformly random string in $\{0,1\}^{L_{\bar{C}}}$. As blindness holds for every $\mathsf{garb.gk}$ and every input $\mathbf{A}_S$, this holds even conditioned on $\widetilde{\mathbf{A}}_S$ and all other values in the hybrids. Hence, the lemma holds. □

Lemma 4.23. $\Pr[1 \leftarrow \mathbf{Hyb}_9^{\mathcal{A}}(1^\lambda)] = \Pr[1 \leftarrow \mathbf{Hyb}_{10}^{\mathcal{A}}(1^\lambda)]$.

Proof. In $\mathbf{Hyb}_9^{\mathcal{A}}(1^\lambda)$, $\widetilde{C}^*$ is uniformly random and is only used to compute $y^* = T^* \oplus \widetilde{C}^*$. So, y^* is uniformly random and independent of T^* and all other values sampled before the challenge phase. Thus, we can sample y^* and $\mathsf{prf.key} \leftarrow \mathsf{SEPRF.Sim}_2(\mathsf{eqKey}, y^*)$ during the setup phase. Moreover, for each query, $t_i \neq t^*$ unless the challenger aborts. So, $\mathsf{SEPRF.Eval}(\mathsf{prf.key}, t_i) = \mathsf{SEPRF.Eval}(\mathsf{eqKey}, t_i)$ by correctness of SEPRF. Hence, the output distributions of $\mathbf{Hyb}_9^{\mathcal{A}}(1^\lambda)$ and $\mathbf{Hyb}_{10}^{\mathcal{A}}(1^\lambda)$ are identical and the lemma holds. □

Lemma 4.24. If SEPRF satisfies security property (Definition 4.2), then for any PPT admissible adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for any $\lambda \in \mathbb{N}$, $|\Pr[1 \leftarrow \mathbf{Hyb}_{10}^{\mathcal{A}}(1^\lambda)] - \Pr[1 \leftarrow \mathbf{Hyb}_{11}^{\mathcal{A}}(1^\lambda)]| \le \mathsf{negl}(\lambda)$.

Proof. This is similar to the proof of Lemma 4.15. The reduction adversary $\mathcal{B}$ samples $t^* \leftarrow \{0,1\}^\lambda$, sends $(1^{L_{\bar{C}}}, t^*)$ to the SEPRF challenger, and receives key. It samples $\mathbf{A}, \mathsf{garb.gk}$, and $T^* \leftarrow \{0,1\}^{L_{\bar{C}}}$ on its own, answers each query using key in place of $\mathsf{prf.key}$, and sends $\mathsf{sk}_S := (\mathsf{key}, \widetilde{\mathbf{A}}_S)$ during the challenge phase. If $\mathsf{key} \leftarrow \mathsf{SEPRF.Sim}_2(\mathsf{eqKey}, y^*)$ for a uniformly random y^*, then $\mathcal{B}$ behaves like $\mathbf{Hyb}_{10}^{\mathcal{A}}(1^\lambda)$. Otherwise, $\mathsf{key} \leftarrow \mathsf{SEPRF.Setup}(1^\lambda, 1^{L_{\bar{C}}})$ and $\mathcal{B}$ behaves like $\mathbf{Hyb}_{11}^{\mathcal{A}}(1^\lambda)$. □

Lemma 4.25. For any PPT adversary $\mathcal{A}$, $|\Pr[1 \leftarrow \mathbf{Hyb}_{11}^{\mathcal{A}}(1^\lambda)] - \Pr[1 \leftarrow \mathbf{Hyb}_{12}^{\mathcal{A}}(1^\lambda)]| \le Q \cdot 2^{-\lambda}$.

Proof. This is similar to the proof of Lemma 4.14. □

Using the above lemmas, we prove the theorem. □

4.6 Payload Hiding Analysis

Theorem 4.26 (Payload Hiding)**.** If SEPRF satisfies security property (Definition 4.2), Garb satisfies security and blindness properties (Definition 4.4), and LWE assumption holds (Assumption 3.4), then Construction 4.6 satisfies payload hiding property (Definition 4.1).

Proof. We use the same sequence of hybrids $\mathbf{Hyb}_0^{\mathcal{A}}(1^\lambda), \ldots, \mathbf{Hyb}_{12}^{\mathcal{A}}(1^\lambda)$ as in the proof of Theorem 4.13 with the following changes. In every hybrid, $\mathcal{A}$ sends (N, i^*, μ_{i^*}) during the setup phase and the challenger uses this μ_{i^*} instead of sampling it. Moreover, $\mathcal{A}$ sends $S \subseteq [N]$ such that $i^* \notin S$ during the challenge phase. So, $\mathbf{Hyb}_0^{\mathcal{A}}(1^\lambda)$ and $\mathbf{Hyb}_{12}^{\mathcal{A}}(1^\lambda)$ are the payload hiding experiments with $b = 1$ and $b = 0$ respectively. In addition, $\mathbf{Hyb}_5^{\mathcal{A}}(1^\lambda)$, $\mathbf{Hyb}_6^{\mathcal{A}}(1^\lambda)$, and $\mathbf{Hyb}_8^{\mathcal{A}}(1^\lambda)$ are modified as follows.

$\mathbf{Hyb}_5^{\mathcal{A}}(1^\lambda)$**.** This is the same as $\mathbf{Hyb}_4^{\mathcal{A}}(1^\lambda)$ except that during the challenge phase, we compute $\mathbf{H}_{S,i^*} = \mathsf{EvalFX}(\mathbf{A}, f_S, \mathsf{encvec}(i^*))$, sample $\mathbf{e_P} \leftarrow \mathcal{D}_{\mathbb{Z},\chi}^{\lambda}$, and set $\mathbf{d}^\top = \mathbf{c}^\top \mathbf{H}_{S,i^*} \mathbf{G}^{-1}(\mathbf{P}) - (\mathbf{s}^\top \mathbf{P} + \mathbf{e}_{\mathbf{P}}^\top) + \widetilde{\mathbf{e}}^\top + \mathsf{tg}$.

$\mathbf{Hyb}_6^{\mathcal{A}}(1^\lambda)$**.** This is the same as $\mathbf{Hyb}_5^{\mathcal{A}}(1^\lambda)$ except that we sample $\mathbf{c}^\top \leftarrow \mathbb{Z}_q^{1\times(\lceil \log N \rceil \cdot m)}$ and $\mathbf{u}^\top \leftarrow \mathbb{Z}_q^{1\times\lambda}$ uniformly and set $\mathbf{d}^\top = \mathbf{c}^\top \mathbf{H}_{S,i^*} \mathbf{G}^{-1}(\mathbf{P}) - \mathbf{u}^\top + \widetilde{\mathbf{e}}^\top + \mathsf{tg}$ during the challenge phase.

$\mathbf{Hyb}_8^{\mathcal{A}}(1^\lambda)$**.** This is the same as $\mathbf{Hyb}_7^{\mathcal{A}}(1^\lambda)$ except that we sample $\mathbf{d}^\top \leftarrow \mathbb{Z}_q^{1\times\lambda}$ and $h \leftarrow \{0,1\}^\lambda$ uniformly during the challenge phase. Note that we no longer need to sample $\mathbf{u}^\top, \widetilde{\mathbf{e}}, \mathsf{tg}$.

Indistinguishability of Hybrids. The indistinguishability of $\mathbf{Hyb}_j^{\mathcal{A}}(1^\lambda)$ and $\mathbf{Hyb}_{j+1}^{\mathcal{A}}(1^\lambda)$ for $j \in \{0, 1, 2, 3, 6, 8, 9, 10, 11\}$ follows similarly to the proofs of Lemma 4.14, Lemma 4.15, Lemma 4.16, Lemma 4.17, Lemma 4.20, Lemma 4.22, Lemma 4.23, Lemma 4.24, and Lemma 4.25 respectively. Note that these reductions do not use the target message μ_{i^*} in any way other than constructing the target circuit, so they work equally well when μ_{i^*} is chosen by $\mathcal{A}$. We argue the remaining transitions below.

Lemma 4.27. For any PPT admissible adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for any $\lambda \in \mathbb{N}$, $|\Pr[1 \leftarrow \mathbf{Hyb}_4^{\mathcal{A}}(1^\lambda)] - \Pr[1 \leftarrow \mathbf{Hyb}_5^{\mathcal{A}}(1^\lambda)]| \leq \mathsf{negl}(\lambda)$.

Proof. As $f_S(i^*) = 0$ by admissibility of $\mathcal{A}$, $\mathbf{c}^\top \mathbf{H}_{S,i^*} \mathbf{G}^{-1}(\mathbf{P}) = \mathbf{s}^\top(\mathbf{A}_S \mathbf{G}^{-1}(\mathbf{P})) + \mathbf{e}^\top \mathbf{H}_{S,i^*} \mathbf{G}^{-1}(\mathbf{P})$. So, $\mathbf{d}^\top$ in $\mathbf{Hyb}_5^{\mathcal{A}}(1^\lambda)$ is equal to $\mathbf{d}^\top$ in $\mathbf{Hyb}_4^{\mathcal{A}}(1^\lambda)$ shifted by $\mathbf{e}^\top \mathbf{H}_{S,i^*} \mathbf{G}^{-1}(\mathbf{P}) - \mathbf{e}_{\mathbf{P}}^\top$. With overwhelming probability, $\left\|\mathbf{e}^\top \mathbf{H}_{S,i^*} \mathbf{G}^{-1}(\mathbf{P}) - \mathbf{e}_{\mathbf{P}}^\top\right\| \leq 2\sqrt{m}\lceil \log N \rceil \chi m^{O(d_f)}$. The rest of the proof is similar to the proof of Lemma 4.18. □

Lemma 4.28. If LWE assumption holds (Assumption 3.4), then for any PPT admissible adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for any $\lambda \in \mathbb{N}$, $|\Pr[1 \leftarrow \mathbf{Hyb}_5^{\mathcal{A}}(1^\lambda)] - \Pr[1 \leftarrow \mathbf{Hyb}_6^{\mathcal{A}}(1^\lambda)]| \leq \mathsf{negl}(\lambda)$.

Proof. Note that in $\mathbf{Hyb}_5^{\mathcal{A}}(1^\lambda)$, $\mathbf{s}, \mathbf{e}, \mathbf{e_P}$ are only used to compute $\mathbf{c}^\top = \mathbf{s}^\top(\mathbf{A} - \mathsf{encvec}(i^*) \otimes \mathbf{G}) + \mathbf{e}^\top$ and $\mathbf{s}^\top \mathbf{P} + \mathbf{e}_{\mathbf{P}}^\top$. This is similar to the proof of Lemma 4.19 except that the reduction adversary $\mathcal{B}$ uses an LWE challenge $(\mathbf{M}, \mathbf{w}^\top)$ with $\lceil \log N \rceil \cdot m + \lambda$ samples. $\mathcal{B}$ parses $\mathbf{M} = [\mathbf{M}_1 \mid \mathbf{M}_2]$ and $\mathbf{w}^\top = [\mathbf{w}_1^\top \mid \mathbf{w}_2^\top]$ where $\mathbf{M}_1 \in \mathbb{Z}_q^{n\times(\lceil \log N \rceil \cdot m)}$ and $\mathbf{M}_2 \in \mathbb{Z}_q^{n\times\lambda}$. It sets $\mathbf{A} := \mathbf{M}_1 + \mathsf{encvec}(i^*) \otimes \mathbf{G}$ during the setup phase. During the challenge phase, it sets $\mathbf{P} := \mathbf{M}_2$, $\mathbf{c}^\top := \mathbf{w}_1^\top$, and computes

$\mathbf{d}^\top = \mathbf{c}^\top \mathbf{H}_{S,i^*} \mathbf{G}^{-1}(\mathbf{P}) - \mathbf{w}_2^\top + \widetilde{\mathbf{e}}^\top + \mathsf{tg}$. Note that $\mathbf{M}_2$ is uniformly random and is not used before the challenge phase. If $\mathbf{w}^\top = \mathbf{s}^\top \mathbf{M} + [\mathbf{e}^\top \mid \mathbf{e}_{\mathbf{P}}^\top]$, then $\mathcal{B}$ behaves like $\mathbf{Hyb}_5^{\mathcal{A}}(1^\lambda)$. Otherwise, $\mathcal{B}$ behaves like $\mathbf{Hyb}_6^{\mathcal{A}}(1^\lambda)$. □

Lemma 4.29. For any PPT admissible adversary $\mathcal{A}$, $|\Pr[1 \leftarrow \mathbf{Hyb}_7^{\mathcal{A}}(1^\lambda)] - \Pr[1 \leftarrow \mathbf{Hyb}_8^{\mathcal{A}}(1^\lambda)]| \leq \lambda/q$.

Proof. In $\mathbf{Hyb}_7^{\mathcal{A}}(1^\lambda)$, $\mathbf{u}^\top$ is uniformly random and is only used to compute $\mathbf{d}^\top = \mathbf{c}^\top \mathbf{H}_{S,i^*} \mathbf{G}^{-1}(\mathbf{P}) - \mathbf{u}^\top + \widetilde{\mathbf{e}}^\top + \mathsf{tg}$. So, $\mathbf{d}^\top$ is uniformly random in $\mathbb{Z}_q^{1\times\lambda}$ and independent of tg and all other values. Moreover, tg is uniformly random in $\mathbb{Z}_q^\lambda$ and is now only used to compute $h = \lceil \mathsf{tg} \rfloor_2 \oplus \mu_{i^*}$. For a uniformly random $x \leftarrow \mathbb{Z}_q$, the number of elements of $\mathbb{Z}_q$ that round to 0 and to 1 differ by at most 1 as q is an odd prime. So, the statistical distance between $\lceil x \rfloor_2$ and a uniformly random bit is at most $1/q$. As the λ entries of tg are independent, the statistical distance between $\lceil \mathsf{tg} \rfloor_2$ and a uniformly random string in $\{0,1\}^\lambda$ is at most λ/q. Thus, $h = \lceil \mathsf{tg} \rfloor_2 \oplus \mu_{i^*}$ is within statistical distance λ/q of a uniformly random string in $\{0,1\}^\lambda$ which is independent of $(\mathbf{c}^\top, \mathbf{P}, \mathbf{d}^\top)$ and all other values, even though μ_{i^*} is chosen by $\mathcal{A}$. This is negligible as $q > \lambda^{\omega(1)} \cdot \lambda \cdot (4B+2)$. Hence, the lemma holds. □

Using the above lemmas, we prove the theorem. □

Corollary 4.30. Assuming the hardness of LWE with sub-exponential modulus-to-noise ratio, for any $N \in \mathbb{N}$, there exists a programmable pseudorandom set sampler scheme for N indices.

4.7 A More Efficient Construction for Polynomially Bounded Users

PPS for bounded polynomial N. The most expensive steps in Construction 4.6 are the computation of the digest $\mathbf{A}_S = \mathsf{EvalF}(\mathbf{A}, f_S)$ in KGen and the computation of $\mathbf{H}_{S,i}$ in Decode. Both of these algorithms homomorphically evaluate the set membership circuit f_S gate by gate, where each gate requires a bit decomposition followed by a matrix multiplication, and $\|\mathbf{H}_{S,i}\|$ grows as $m^{O(d_f)}$. When N is polynomially bounded, we can avoid this homomorphic evaluation by replacing f_S with an inner product. More specifically, we encode an index $i \in [N]$ as the unit vector $\mathbb{1}_{N,i} \in \{0,1\}^N$ and a set $S \subseteq [N]$ as its indicator vector $\mathbf{y}_S \in \{0,1\}^N$ so that $f_S(i) = \mathbb{1}_{N,i}^\top \mathbf{y}_S$. Let $\mathbf{V}_N = \mathsf{MCOM.Verify}(1^{Nm})$ be the verification matrix of the matrix commitment scheme of [Wee25] and let $\mathbf{B}_1 \leftarrow \mathbb{Z}_q^{n\times m}$ be part of the crs. Consider the expanded matrix $\mathbf{A} = -\mathbf{B}_1 \mathbf{V}_N = [\mathbf{A}_1 \mid \ldots \mid \mathbf{A}_N]$ where each $\mathbf{A}_j \in \mathbb{Z}_q^{n\times m}$. The BGG+ evaluation [BGG+14] of the above linear function is simply $\mathbf{A}_S = \mathbf{A}(\mathbf{y}_S \otimes \mathbf{I}_m) = \sum_{j\in S} \mathbf{A}_j$. That is, KGen only needs to add $|S|$ blocks of $\mathbf{A}$, which is a deterministic function of the crs and can be computed once and stored.

As the encoded index $\mathbb{1}_{N,i}$ has N bits, the BGG+ encoding $\mathbf{s}^\top(\mathbf{A} - \mathbb{1}_{N,i}^\top \otimes \mathbf{G}) + \mathbf{e}^\top$ would make the PPS ciphertexts linear in N. We can use the matrix commitment scheme of [Wee25] to make this compact. It is easy to check that Decode can use $-\sum_{j\in S} \mathbf{Z}_{i,j}$ and $-\sum_{j\in S} \mathbf{V}_{N,j}$ to compute the homomorphic computation where $\mathbf{Z}$ is the opening matrix [Wee25]. Using $2m^2$-decomposed LWE assumption [AMR25], we can build matrix commitment schemes with a public-coin crs [Wee25, WW25] to get public-coin in PPS.

This construction retains full compactness, i.e., $|\mathsf{crs}|, |\mathsf{msk}|, |\mathsf{sk}_S|, |\mathsf{ct}| = \mathsf{poly}(\lambda, \log N)$. However, some running times grow with N. So, this construction is only useful when N is polynomially bounded and in particular, it cannot replace the PPS scheme from Construction 4.6 in the FBE compiler from Construction 5.3 where the indices are λ-bit tags.

5 Application I: Broadcast Encryption

In this section, we show that PPS directly yields adaptively secure flexible broadcast encryption (FBE) in the standard model. Our compiler (Construction 5.3) generalizes the two-key compiler of Gentry and Waters [GW09, FWW23], which upgrades semi-static security to adaptive security. The proof is more involved than that of [GW09, FWW23]. The challenger must program each user's PPS ciphertext when the user registers. At that point, it does not know whether the user will be in the challenge set or will be corrupted. We therefore switch users one at a time and use a pebbling argument [JKK+17, KW20] (Lemma 5.7). We start by with defining FBE.

Syntax. A Flexible Broadcast Encryption scheme (FBE) consists of the following polynomial-time algorithms.

$\mathsf{GSetup}(1^\lambda) \to \mathsf{crs}$**.** The possibly randomized *public-coin* global setup algorithm takes as input security parameter λ and outputs common random string crs. The following algorithms take crs implicitly.

$\mathsf{KGen} \to (\mathsf{pk}, \mathsf{sk})$**.** The randomized key generation algorithm outputs public-secret key pair $(\mathsf{pk}, \mathsf{sk})$.

$\mathsf{Enc}((\mathsf{pk}_1, \ldots, \mathsf{pk}_N), \mu) \to \mathsf{ct}$**.** The randomized encryption algorithm takes as input N public keys $\mathsf{pk}_1, \ldots, \mathsf{pk}_N$, message $\mu \in \{0,1\}$, and outputs a ciphertext ct.

$\mathsf{Dec}((\mathsf{pk}_1, \ldots, \mathsf{pk}_N), (i^*, \mathsf{sk}_{i^*}), \mathsf{ct}) \to \mu'$ The deterministic decryption algorithm takes as input N public keys $(\mathsf{pk}_1, \ldots, \mathsf{pk}_N)$, secret key for $i^* \in [N]$, sk_{i^*}, ciphertext ct, and outputs $\mu' \in \{0,1\} \cup \{\perp\}$.

Definition 5.1 (FBE)**.** A FBE scheme $(\mathsf{GSetup}, \mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$ is said to be a FBE scheme if it satisfies the following properties.

Correctness. For any $\lambda \in \mathbb{N}, N = N(\lambda)$, any $i^* \in [N], \mu \in \{0,1\}$,

$$\Pr\left[\mu = \mathsf{Dec}((\mathsf{pk}_1, \ldots, \mathsf{pk}_N), (i^*, \mathsf{sk}_{i^*}), \mathsf{ct}) \;:\; \begin{array}{l} \mathsf{crs} \leftarrow \mathsf{GSetup}(1^\lambda), \\ \forall\, j \in [N], (\mathsf{pk}_j, \mathsf{sk}_j) \leftarrow \mathsf{KGen}, \\ \mathsf{ct} \leftarrow \mathsf{Enc}((\mathsf{pk}_1, \ldots, \mathsf{pk}_N), \mu) \end{array}\right] = 1$$

Compactness. There exists a polynomial $\mathsf{poly}(\cdot,\cdot)$ such that for any $\lambda \in \mathbb{N}, N \leq 2^\lambda$, $|\mathsf{ct}| = o(N) \cdot \mathsf{poly}_{\mathsf{ct}}(\lambda, \log N)$.

Adaptive Security. For any stateful admissible PPT adversary $\mathcal{A}$, there exists a negligible function such that $\forall\ \lambda \in \mathbb{N}$,

$$\left|\Pr\left[b \leftarrow \mathcal{A}(\mathsf{ct}) \;:\; \begin{array}{l} \mathcal{K} := \emptyset, \mathsf{ctr} := 1 \\ \mathsf{crs} \leftarrow \mathsf{GSetup}(1^\lambda), \\ S \leftarrow \mathcal{A}^{\mathcal{O}(\cdot,\cdot)}(\mathsf{crs}), b \leftarrow \{0,1\}, \\ \mathsf{ct} \leftarrow \mathsf{Enc}((\mathsf{pk}_1, \ldots, \mathsf{pk}_N), b) \end{array}\right] - \frac{1}{2}\right| \leq \mathsf{negl}(\lambda)$$

Here, $(\mathsf{pk}_1, \ldots, \mathsf{pk}_N)$ are derived as follows: let $(h_1, \ldots, h_N)$ be lexicographical ordering of S with each $h_i \in [\mathsf{ctr}]$ for $i \in [N]$. Then, $\mathsf{pk}_i = \mathsf{pk}_{h_i}$. $\mathcal{O}$ is a stateful oracle that responds to the following queries.

$(\mathsf{KGen}, \perp)$. $\mathcal{A}$ queries for key generation. $\mathcal{O}$ samples $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KGen}$, updates $\mathcal{K} := \mathcal{K} \cup \{(\mathsf{ctr}, \mathsf{pk}, \mathsf{sk})\}$ and sends $(\mathsf{ctr}, \mathsf{pk})$ to $\mathcal{A}$. Then, $\mathcal{O}$ updates $\mathsf{ctr} := \mathsf{ctr} + 1$.

(Corr, h'). $\mathcal{A}$ queries for key corruption for handle $h' \in [\mathsf{ctr}]$. If there is no entry for $(h', \mathsf{pk}_{h'}, \mathsf{sk}_{h'})$ in $\mathcal{K}$ send $\perp$ to $\mathcal{A}$. Otherwise, $\mathcal{O}$ responds with $\mathsf{sk}_{h'}$ to $\mathcal{A}$. $\mathcal{O}$ also updates $\mathcal{K} := \mathcal{K} \setminus \{(h', \mathsf{pk}_{h'}, \mathsf{sk}_{h'})\}$.

A stateful PPT machine $\mathcal{A}$ is said to be admissible if $S \subseteq [\mathsf{ctr}]$, for each $h^* \in S$, there exists an entry $(h^*, \mathsf{pk}_{h^*}, \mathsf{sk}_{h^*}) \in \mathcal{K}$.

Definition 5.2 (Semi-Static Security)**.** An FBE scheme $(\mathsf{GSetup}, \mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$ is said to be a semi-statically secure FBE scheme if satisfies correctness and compactness of Definition 5.1 and semi-static security which is the same as adaptive security except that $\mathcal{A}$ is no longer allowed any (Corr, h') queries.

We now show how to use a PPS scheme for $\mathcal{M} = \{0,1\}$ to upgrade any semi-statically secure FBE scheme to an adaptively-secure FBE scheme in the standard model without any significant overhead to the parameters.

Construction 5.3. We construct an adaptively secure FBE scheme using a semi-statically secure FBE scheme SSFBE (Definition 5.2) and a PPS scheme for $\mathcal{M} = \{0,1\}$ (Definition 4.1) below.

$\mathsf{GSetup}(1^\lambda)$. Sample $\mathsf{fbe.crs} \leftarrow \mathsf{SSFBE.GSetup}(1^\lambda)$ and $\mathsf{pps.crs} \leftarrow \mathsf{GSetup}(1^\lambda, 2^\lambda)$.

Output $\mathsf{crs} := (\mathsf{fbe.crs}, \mathsf{pps.crs})$.

KGen. Do the following.

1. Sample $(\mathsf{fbe.pk}_0, \mathsf{fbe.sk}_0) \leftarrow \mathsf{SSFBE.KGen}$ and $(\mathsf{fbe.pk}_1, \mathsf{fbe.sk}_1) \leftarrow \mathsf{SSFBE.KGen}$. Sample bit $s \leftarrow \{0,1\}$.
2. Sample tag $\tau \leftarrow \{0,1\}^\lambda$ and $\mathsf{pps.ct} \leftarrow \mathsf{PPS.pkEnc}(\tau)$.

Set $\mathsf{pk} := (\mathsf{fbe.pk}_0, \mathsf{fbe.pk}_1, \tau, \mathsf{pps.ct})$ and $\mathsf{sk} := (s, \mathsf{fbe.sk}_s)$.

$\mathsf{Enc}((\mathsf{pk}_1, \ldots, \mathsf{pk}_N), \mu)$. Parse pk_i as $(\mathsf{fbe.pk}_{i,0}, \mathsf{fbe.pk}_{i,1}, \tau_i, \mathsf{pps.ct}_i)$ $\forall\ i \in [N]$. Do the following.

1. Compute set of tags $S = \{\tau_i\}_{i \in [N]}$.
2. Sample $\mathsf{pps.msk} \leftarrow \mathsf{PPS.MSKGen}$, $\mathsf{pps.sk}_S \leftarrow \mathsf{PPS.KGen}(\mathsf{pps.msk}, S)$.
3. Compute $t_i = \mathsf{PPS.Decode}(S, \mathsf{pps.sk}_S, \tau_i, \mathsf{pps.ct}_i)$ for each $i \in [N]$.
4. Sample ciphertexts

$$\mathsf{fbe.ct}_0 \leftarrow \mathsf{SSFBE.Enc}((\mathsf{fbe.pk}_{i,t_i})_{i \in [N]}, \mu)$$
$$\mathsf{fbe.ct}_1 \leftarrow \mathsf{SSFBE.Enc}((\mathsf{fbe.pk}_{i,1 \oplus t_i})_{i \in [N]}, \mu)$$

Output $\mathsf{ct} := (\mathsf{pps.sk}_S, \mathsf{fbe.ct}_0, \mathsf{fbe.ct}_1)$

$\mathsf{Dec}((\mathsf{pk}_1, \ldots, \mathsf{pk}_N), i^*, \mathsf{sk}_{i^*}, \mathsf{ct})$. Parse pk_i as $(\mathsf{fbe.pk}_{i,0}, \mathsf{fbe.pk}_{i,1}, \tau_i, \mathsf{pps.ct}_i)$, sk_{i^*} as $(s^*, \mathsf{fbe.sk}_{i^*, s^*})$, and ct as $(\mathsf{pps.sk}_S, \mathsf{fbe.ct}_0, \mathsf{fbe.ct}_1)$. Do the following.

1. Compute set of tags $S = \{\tau_i\}_{i \in [N]}$.

2. Compute $t_i = \mathsf{PPS.Decode}(S, \mathsf{pps.sk}_S, \tau_i, \mathsf{pps.ct}_i)$ for each $i \in [N]$.
3. If $s^* = t_{i^*}$, compute $\mu' = \mathsf{SSFBE.Dec}((\mathsf{fbe.pk}_{i,t_i})_{i\in[N]}, i^*, \mathsf{fbe.sk}_{i^*,s^*}, \mathsf{fbe.ct}_0)$.
4. Otherwise, compute $\mu' = \mathsf{SSFBE.Dec}((\mathsf{fbe.pk}_{i,1\oplus t_i})_{i\in[N]}, i^*, \mathsf{fbe.sk}_{i^*,s^*}, \mathsf{fbe.ct}_1)$.

Output μ'

Correctness. As Enc and Dec receive the same set of public keys $(\mathsf{pk}_1, \ldots, \mathsf{pk}_N)$, the set S computed in Step 1 of both of these algorithms is the same. Moreover, as $\mathsf{Enc}, \mathsf{Dec}$ uses the $\mathsf{pps.sk}_S$ computed in Enc, each t_i will also be the same by correctness property of PPS.

Thus, if $s^* = t_i$, the $\mathsf{fbe.ct}_0$ contains the key $\mathsf{fbe.pk}_{i^*,t_{i^*}} = \mathsf{fbe.pk}_{i^*,s^*}$. So, by correctness of SSFBE, $\mathsf{fbe.ct}_0$ can be decrypted using $(\mathsf{fbe.pk}_{i,t_i})_{i\in[N]}$ and $\mathsf{fbe.sk}_{i^*,s^*}$. The other case, follows analogously and thus, $\mu' = \mu$ as long as SSFBE correctness holds.

Compactness. Firstly, it is clear that $|\mathsf{ct}| = 2|\mathsf{fbe.ct}| + |\mathsf{pps.sk}_S| = 2o(N)\mathsf{poly}(\lambda, \log N) + \mathsf{poly}(\lambda) = o(N)\mathsf{poly}(\lambda, \log N)$. Thus, compactness holds for the construction.

Moreover, the $|\mathsf{crs}| = |\mathsf{fbe.crs}| + |\mathsf{pps.crs}| = |\mathsf{fbe.crs}| + \mathsf{poly}(\lambda)$, $|\mathsf{pk}| = 2|\mathsf{fbe.pk}| + \lambda + |\mathsf{pps.ct}| = 2|\mathsf{fbe.pk}| + \mathsf{poly}(\lambda)$, and $|\mathsf{sk}| = 1 + |\mathsf{fbe.sk}|$. Thus, our compiler blows up each parameter at most by $2\times$ and adds at most $\mathsf{poly}(\lambda)$.

5.1 Security Analysis

Theorem 5.4. If SSFBE is a satisfies semi-static security property (Definition 5.2) and PPS is a PPS scheme for message space $\{0,1\}$, then Construction 5.3 is an adaptively secure FBE scheme (Definition 5.1).

Proof. We prove this theorem using the following hybrids for an admissible adversary.

$\mathbf{Hyb}_0^{\mathcal{A}}(1^\lambda)$. This is the adaptive security experiment (Definition 5.1) with $b = 0$.

Setup Phase. Sample $\mathsf{fbe.crs} \leftarrow \mathsf{SSFBE.GSetup}(1^\lambda)$ and $\mathsf{pps.crs} \leftarrow \mathsf{PPS.GSetup}(1^\lambda, 2^\lambda)$. Set $\mathsf{crs} := (\mathsf{fbe.crs}, \mathsf{pps.crs})$ and send crs to $\mathcal{A}$. Initiate key dictionary $\mathcal{K} := \emptyset$ and counter $\mathsf{ctr} := 1$.

Key Generation Query. Do the following.

1. Sample $(\mathsf{fbe.pk}_0, \mathsf{fbe.sk}_0) \leftarrow \mathsf{SSFBE.KGen}$ and $(\mathsf{fbe.pk}_1, \mathsf{fbe.sk}_1) \leftarrow \mathsf{SSFBE.KGen}$. Sample bit $s \leftarrow \{0,1\}$.
2. Sample tag $\tau \leftarrow \{0,1\}^\lambda$ and $\mathsf{pps.ct} \leftarrow \mathsf{PPS.pkEnc}(\tau)$.

Add $(\mathsf{ctr}, \mathsf{pk} := (\mathsf{fbe.pk}_0, \mathsf{fbe.pk}_1, \tau, \mathsf{pps.ct}), \mathsf{sk} := (s, \mathsf{fbe.sk}_s))$ to $\mathcal{K}$. Send $(\mathsf{ctr}, \mathsf{pk})$ to $\mathcal{A}$. Increment $\mathsf{ctr} := \mathsf{ctr} + 1$.

Key Corruption Query. Given $h' \in [\mathsf{ctr}]$, check if there is an entry for $(h', \mathsf{pk}_{h'}, \mathsf{sk}_{h'})$ in $\mathcal{K}$. If not, output $\perp$. Otherwise, send $\mathsf{sk}_{h'}$ to $\mathcal{A}$.

Challenge Encryption. Given S which lexicographically parses as $(h_1, \ldots, h_N)$ for some $N = |S| = \mathsf{poly}(\lambda)$, retrieve $(h_i, \mathsf{pk}_i, \mathsf{sk}_i)$ from $\mathcal{K}$. If retrieval fails for any entry, abort and output 0. Otherwise,

1. Parse pk_i as $(\mathsf{fbe.pk}_{i,0}, \mathsf{fbe.pk}_{i,1}, \tau_i, \mathsf{pps.ct}_i)$ for each $i \in [N]$.

2. Compute set of tags $S = \{\tau_i\}_{i\in[N]}$.
3. Sample $\mathsf{pps.msk} \leftarrow \mathsf{PPS.MSKGen}$, $\mathsf{pps.sk}_S \leftarrow \mathsf{PPS.KGen}(\mathsf{pps.msk}, S)$.
4. Compute $t_i = \mathsf{PPS.Decode}(S, \mathsf{pps.sk}_S, \tau_i, \mathsf{pps.ct}_i)$ for each $i \in [N]$.
5. Sample ciphertexts

$$\mathsf{fbe.ct}_0 \leftarrow \mathsf{SSFBE.Enc}((\mathsf{fbe.pk}_{i,t_i})_{i\in[N]}, 0)$$
$$\mathsf{fbe.ct}_1 \leftarrow \mathsf{SSFBE.Enc}((\mathsf{fbe.pk}_{i,1\oplus t_i})_{i\in[N]}, 0)$$

Send $\mathsf{ct} := (\mathsf{pps.sk}_S, \mathsf{fbe.ct}_0, \mathsf{fbe.ct}_1)$ to $\mathcal{A}$.

Guess Phase. $\mathcal{A}$ outputs guess b'. Output 1 if $b = b'$ and 0 otherwise.

$\mathbf{Hyb}_1^{\mathcal{A}}(1^\lambda)$. This is the same as $\mathbf{Hyb}_0^{\mathcal{A}}(1^\lambda)$ except that we sample $\mathsf{pps.msk}$ during setup phase. And, during key generation phase, if any τ is repeated, the challenger aborts and outputs 0.

$\mathbf{Hyb}_{2,j}^{\mathcal{A}}(1^\lambda)$ for $j \in [Q+1]$. Here Q is a bound on the number of key generation queries $\mathcal{A}$ will make. This is the same as $\mathbf{Hyb}_1^{\mathcal{A}}(1^\lambda)$ except that in the first $j-1$ key generation queries made by $\mathcal{A}$, we will sample $\mathsf{pps.ct}_\tau \leftarrow \mathsf{PPS.skEnc}(\mathsf{pps.msk}, \tau, 1 \oplus s)$ where s is the random bit sampled during key generation.

Setup Phase. Sample $\mathsf{fbe.crs} \leftarrow \mathsf{SSFBE.GSetup}(1^\lambda)$ and $\mathsf{pps.crs} \leftarrow \mathsf{PPS.GSetup}(1^\lambda, 2^\lambda)$. Set $\mathsf{crs} := (\mathsf{fbe.crs}, \mathsf{pps.crs})$ and send crs to $\mathcal{A}$. Sample $\mathsf{pps.msk} \leftarrow \mathsf{PPS.MSKGen}$. Initiate key dictionary $\mathcal{K} := \emptyset$ and counter $\mathsf{ctr} := 1$.

Key Generation Query. Do the following.

1. Sample $(\mathsf{fbe.pk}_0, \mathsf{fbe.sk}_0) \leftarrow \mathsf{SSFBE.KGen}$ and $(\mathsf{fbe.pk}_1, \mathsf{fbe.sk}_1) \leftarrow \mathsf{SSFBE.KGen}$. Sample bit $s \leftarrow \{0,1\}$.
2. Sample $\tau \leftarrow \{0,1\}^\lambda$. If $\mathsf{ctr} < j$, $\mathsf{pps.ct}_\tau \leftarrow \mathsf{PPS.skEnc}(\mathsf{pps.msk}, \tau, 1 \oplus s)$. Otherwise, $\mathsf{pps.ct} \leftarrow \mathsf{PPS.pkEnc}(\tau)$.

Add $(\mathsf{ctr}, \mathsf{pk} := (\mathsf{fbe.pk}_0, \mathsf{fbe.pk}_1, \tau, \mathsf{pps.ct}), \mathsf{sk} := (s, \mathsf{fbe.sk}_s))$ to $\mathcal{K}$. Send $(\mathsf{ctr}, \mathsf{pk})$ to $\mathcal{A}$. Increment $\mathsf{ctr} := \mathsf{ctr} + 1$.

Key Corruption, Challenge Encryption, Guess Phase. Same as $\mathbf{Hyb}_1^{\mathcal{A}}(1^\lambda)$.

$\mathbf{Hyb}_3^{\mathcal{A}}(1^\lambda)$. This is the same as $\mathbf{Hyb}_{2,Q+1}^{\mathcal{A}}(1^\lambda)$. We write the full hybrid for clarity.

Setup Phase. Sample $\mathsf{fbe.crs} \leftarrow \mathsf{SSFBE.GSetup}(1^\lambda)$ and $\mathsf{pps.crs} \leftarrow \mathsf{PPS.GSetup}(1^\lambda, 2^\lambda)$. Set $\mathsf{crs} := (\mathsf{fbe.crs}, \mathsf{pps.crs})$ and send crs to $\mathcal{A}$. Sample $\mathsf{pps.msk} \leftarrow \mathsf{PPS.MSKGen}$. Initiate key dictionary $\mathcal{K} := \emptyset$ and counter $\mathsf{ctr} := 1$.

Key Generation Query. Do the following.

1. Sample $(\mathsf{fbe.pk}_0, \mathsf{fbe.sk}_0) \leftarrow \mathsf{SSFBE.KGen}$ and $(\mathsf{fbe.pk}_1, \mathsf{fbe.sk}_1) \leftarrow \mathsf{SSFBE.KGen}$. Sample bit $s \leftarrow \{0,1\}$.
2. Sample $\tau \leftarrow \{0,1\}^\lambda$ and $\mathsf{pps.ct} \leftarrow \mathsf{PPS.skEnc}(\mathsf{pps.msk}, \tau, 1 \oplus s)$.

Add $(\mathsf{ctr}, \mathsf{pk} := (\mathsf{fbe.pk}_0, \mathsf{fbe.pk}_1, \tau, \mathsf{pps.ct}), \mathsf{sk} := (s, \mathsf{fbe.sk}_s))$ to $\mathcal{K}$. Send $(\mathsf{ctr}, \mathsf{pk})$ to $\mathcal{A}$. Increment $\mathsf{ctr} := \mathsf{ctr} + 1$.

Key Corruption Query. Given $h' \in [\mathsf{ctr}]$, check if there is an entry for $(h', \mathsf{pk}_{h'}, \mathsf{sk}_{h'})$ in $\mathcal{K}$. If not, output $\perp$. Otherwise, send $\mathsf{sk}_{h'}$ to $\mathcal{A}$.

Challenge Encryption. Given S which lexicographically parses as $(h_1, \ldots, h_N)$ for some $N = |S| = \mathsf{poly}(\lambda)$, retrieve $(h_i, \mathsf{pk}_i, \mathsf{sk}_i)$ from $\mathcal{K}$. If retrieval fails for any entry, abort and output 0. Otherwise,

1. Parse pk_i as $(\mathsf{fbe.pk}_{i,0}, \mathsf{fbe.pk}_{i,1}, \tau_i, \mathsf{pps.ct}_i)$ for each $i \in [N]$.
2. Compute $S = \{\tau_i\}_{i \in [N]}$.
3. Sample $\mathsf{pps.sk}_S \leftarrow \mathsf{PPS.KGen}(\mathsf{pps.msk}, S)$.
4. Compute $t_i = \mathsf{PPS.Decode}(S, \mathsf{pps.sk}_S, \tau_i, \mathsf{pps.ct}_i)$ for each $i \in [N]$.
5. Sample ciphertexts

$$\mathsf{fbe.ct}_0 \leftarrow \mathsf{SSFBE.Enc}((\mathsf{fbe.pk}_{i,t_i})_{i \in [N]}, 0)$$
$$\mathsf{fbe.ct}_1 \leftarrow \mathsf{SSFBE.Enc}((\mathsf{fbe.pk}_{i,1 \oplus t_i})_{i \in [N]}, 0)$$

Send $\mathsf{ct} := (\mathsf{pps.sk}_S, \mathsf{fbe.ct}_0, \mathsf{fbe.ct}_1)$ to $\mathcal{A}$.

Guess Phase. $\mathcal{A}$ outputs guess b'. Output 1 if $b = b'$ and 0 otherwise.

$\mathbf{Hyb}_4^{\mathcal{A}}(1^\lambda)$**.** This is the same as $\mathbf{Hyb}_3^{\mathcal{A}}(1^\lambda)$ except that we encrypt 1 in $\mathsf{fbe.ct}_0$ during challenge encryption.

$\mathbf{Hyb}_5^{\mathcal{A}}(1^\lambda)$**.** This is the same as $\mathbf{Hyb}_4^{\mathcal{A}}(1^\lambda)$ except that we sample $\mathsf{pps.ct}_\tau \leftarrow \mathsf{PPS.pkEnc}(\tau)$ in each key generation query.

$\mathbf{Hyb}_6^{\mathcal{A}}(1^\lambda)$**.** This is the same as $\mathbf{Hyb}_5^{\mathcal{A}}(1^\lambda)$ except that we sample $\mathsf{pps.ct}_\tau \leftarrow \mathsf{PPS.skEnc}(\mathsf{pps.msk}, \tau, s)$ where s is the random bit sampled during key generation.

$\mathbf{Hyb}_7^{\mathcal{A}}(1^\lambda)$**.** This is the same as $\mathbf{Hyb}_6^{\mathcal{A}}(1^\lambda)$ except that we encrypt 1 in $\mathsf{fbe.ct}_1$ during challenge encryption.

$\mathbf{Hyb}_8^{\mathcal{A}}(1^\lambda)$**.** This is the adaptive security experiment from Definition 5.1 with $b = 1$. In a bit more detail, the difference between $\mathbf{Hyb}_7^{\mathcal{A}}(1^\lambda)$ and $\mathbf{Hyb}_8^{\mathcal{A}}(1^\lambda)$ is that we sample $\mathsf{pps.ct}_\tau \leftarrow \mathsf{PPS.pkEnc}(\tau)$ in each key generation query and not abort and output 0 when each τ generated during key generation queries are not unique.

Lemma 5.5. For any PPT adversary, there exists a negligible function $\mathsf{negl}(\cdot)$ such that $|\Pr[1 \leftarrow \mathbf{Hyb}_0^{\mathcal{A}}(1^\lambda)] - \Pr[1 \leftarrow \mathbf{Hyb}_1^{\mathcal{A}}(1^\lambda)]| \leq \mathsf{negl}(\lambda)$.

Proof. There are two changes between these hybrids. Firstly, we are sampling $\mathsf{pps.msk}$ early during setup phase. This change will not increase difference in an adversary's advantage.

Secondly, as $\mathcal{A}$ is a PPT machine, there exists a polynomial $Q = Q(\lambda)$ which upper bounds the number of key generation queries $\mathcal{A}$ makes. Let's say the challenger samples $\tau_1, \ldots, \tau_Q \in \{0,1\}^\lambda$

during these key generation queries. As all of these are sampled uniformly randomly, the probability that there will be a collision is at most $Q^2 2^{-\lambda}$ which is a negligible function. So, the probability that in $\mathbf{Hyb}_1^{\mathcal{A}}(1^\lambda)$, challenger aborts and outputs 0 is at most $Q^2 2^{-\lambda}$. Conditioned on that not happening, $\Pr[1 \leftarrow \mathbf{Hyb}_1^{\mathcal{A}}(1^\lambda)]$ is the same as $\Pr[1 \leftarrow \mathbf{Hyb}_0^{\mathcal{A}}(1^\lambda)]$. Thus, the lemma holds. □

Lemma 5.6. $\Pr[1 \leftarrow \mathbf{Hyb}_1^{\mathcal{A}}(1^\lambda)] = \Pr[1 \leftarrow \mathbf{Hyb}_{2,1}^{\mathcal{A}}(1^\lambda)]$.

Proof. In $\mathbf{Hyb}_{2,1}^{\mathcal{A}}(1^\lambda)$ we are not changing any $\mathsf{pps.ct}_\tau$ to be sampled via $\mathsf{PPS.skEnc}$. So, the output distribution of these two hybrids are exactly the same and thus the lemma holds. □

Lemma 5.7. If PPS satisfies mode indistinguishability and payload hiding then for any $j \in [Q]$, for any stateful admissible PPT adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for any $\lambda \in \mathbb{N}$, $|\Pr[1 \leftarrow \mathbf{Hyb}_{2,j}^{\mathcal{A}}(1^\lambda)] - \Pr[1 \leftarrow \mathbf{Hyb}_{2,j+1}^{\mathcal{A}}(1^\lambda)]| \leq \mathsf{negl}(\lambda)$.

Proof. Note that the only difference between these two hybrid experiments is that we are changing the distribution of $\mathsf{pps.ct}$ for the j-th key generation query. In $\mathbf{Hyb}_{2,j}^{\mathcal{A}}(1^\lambda)$, we sample it using $\mathsf{PPS.pkEnc}$ and in $\mathbf{Hyb}_{2,j+1}^{\mathcal{A}}(1^\lambda)$, we sample it using $\mathsf{PPS.skEnc}$. We will prove this lemma using piecewise guessing argument [JKK+17, KW20]. We will guess one bit corresponding to whether the j-th generated key will be in the challenge set or not. That is, let E denote the event that the pk generated in the j-th key is part of the challenge set. Let $\overline{\mathsf{E}}$ denote the complement of this event. We point out that $\overline{\mathsf{E}}$ does not just mean that the j-th key will be corrupted. It just means that it is not part of the challenge key set. Then,

$$\Pr[1 \leftarrow \mathbf{Hyb}_{2,j}^{\mathcal{A}}(1^\lambda)] = \Pr[1 \leftarrow \mathbf{Hyb}_{2,j}^{\mathcal{A}}(1^\lambda) \wedge \mathsf{E}] + \Pr[1 \leftarrow \mathbf{Hyb}_{2,j}^{\mathcal{A}}(1^\lambda) \wedge \overline{\mathsf{E}}]$$
$$\Pr[1 \leftarrow \mathbf{Hyb}_{2,j+1}^{\mathcal{A}}(1^\lambda)] = \Pr[1 \leftarrow \mathbf{Hyb}_{2,j+1}^{\mathcal{A}}(1^\lambda) \wedge \mathsf{E}] + \Pr[1 \leftarrow \mathbf{Hyb}_{2,j+1}^{\mathcal{A}}(1^\lambda) \wedge \overline{\mathsf{E}}]$$

Hence, it enough to show that $|\Pr[1 \leftarrow \mathbf{Hyb}_{2,j}^{\mathcal{A}}(1^\lambda) \wedge \mathsf{E}] - \Pr[1 \leftarrow \mathbf{Hyb}_{2,j+1}^{\mathcal{A}}(1^\lambda) \wedge \mathsf{E}]| \leq \mathsf{negl}(\lambda)$ and $|\Pr[1 \leftarrow \mathbf{Hyb}_{2,j}^{\mathcal{A}}(1^\lambda) \wedge \overline{\mathsf{E}}] - \Pr[1 \leftarrow \mathbf{Hyb}_{2,j+1}^{\mathcal{A}}(1^\lambda) \wedge \overline{\mathsf{E}}]| \leq \mathsf{negl}(\lambda)$ to prove this lemma.

Claim 5.8. Let E be the event from Lemma 5.7. If PPS satisfies mode indistinguishability, then $|\Pr[1 \leftarrow \mathbf{Hyb}_{2,j}^{\mathcal{A}}(1^\lambda) \wedge \mathsf{E}] - \Pr[1 \leftarrow \mathbf{Hyb}_{2,j+1}^{\mathcal{A}}(1^\lambda) \wedge \mathsf{E}]| \leq \mathsf{negl}(\lambda)$.

Proof. As we are guaranteed that event E takes place, we know that the key generated in the j-th key generation query will be part of challenge set. Let $\tau_j \in \{0,1\}^\lambda$ denote the tag generated during the j-th key generation query. Then $\tau_j \in S$ where S is the set of tags computed during challenge query phase. We use the following sub-hybrids to argue this claim.

$\mathbf{SubHyb}_1^{\mathcal{A}}(1^\lambda)$. In this sub-hybrid, we will perform all the tasks similarly to $\mathbf{Hyb}_{2,j}^{\mathcal{A}}(1^\lambda)$ except that for the j-th key generation query, we will not sample the bit s_j used for secret keys.

$\mathbf{SubHyb}_2^{\mathcal{A}}(1^\lambda)$. In this sub-hybrid, we will perform all the tasks similarly to $\mathbf{SubHyb}_1^{\mathcal{A}}(1^\lambda)$ except that we will sample $\mathsf{pps.ct}_j \leftarrow \mathsf{PPS.skEnc}(\mathsf{pps}, \mathsf{msk}, \tau_j, \mu_j^*)$ after sampling $\mu_j^* \leftarrow \{0,1\}$.

$\mathbf{SubHyb}_3^{\mathcal{A}}(1^\lambda)$. This is the same as $\mathbf{SubHyb}_2^{\mathcal{A}}(1^\lambda)$ except that, once we sample $\mathsf{pps.sk}_S \leftarrow \mathsf{PPS.KGen}(\mathsf{pps.msk}, S)$, we will compute $s_j = 1 \oplus \mathsf{PPS.Decode}(S, \mathsf{pps.sk}_S, \tau_j, \mathsf{pps.ct}_j)$.

$\mathbf{SubHyb}_4^{\mathcal{A}}(1^\lambda)$. This is same as $\mathbf{SubHyb}_3^{\mathcal{A}}(1^\lambda)$ except that we will sample $s_j = 1 \oplus \mu_j^*$ right after sampling $\mathsf{pps.ct}_j \leftarrow \mathsf{PPS.skEnc}(\mathsf{pps}, \mathsf{msk}, \tau_j, \mu_j^*)$ during j-th key generation phase.

Proposition 5.9. $\Pr[1 \leftarrow \mathbf{Hyb}_{2,j}^{\mathcal{A}}(1^\lambda) \wedge \mathsf{E}] = \Pr[1 \leftarrow \mathbf{SubHyb}_1^{\mathcal{A}}(1^\lambda)]$.

Proof. As event E happens, by admissibility of $\mathcal{A}$, the j-th will not be corrupted. It means that the bit s_j of the j-th key generation query is no longer be used in the experiment $\mathbf{Hyb}_{2,j}^{\mathcal{A}}(1^\lambda) \wedge \mathsf{E}$. So, leaving it unassigned will not change the view and thus the advantage of $\mathcal{A}$ in any way. So, the quantity $\Pr[1 \leftarrow \mathbf{SubHyb}_1^{\mathcal{A}}(1^\lambda)]$ is same as $\Pr[1 \leftarrow \mathbf{Hyb}_{2,j}^{\mathcal{A}}(1^\lambda) \wedge \mathsf{E}]$. □

Proposition 5.10. If PPS satisfies mode indistinguishability property (Definition 4.1), for any PPT admissible adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that $|\Pr[1 \leftarrow \mathbf{SubHyb}_1^{\mathcal{A}}(1^\lambda)] - \Pr[1 \leftarrow \mathbf{SubHyb}_2^{\mathcal{A}}(1^\lambda)]| \leq \mathsf{negl}(\lambda)$.

Proof. Assume for the sake of contradiction that there exists an adversary $\mathcal{A}$ such that $|\Pr[1 \leftarrow \mathbf{SubHyb}_1^{\mathcal{A}}(1^\lambda)] - \Pr[1 \leftarrow \mathbf{SubHyb}_2^{\mathcal{A}}(1^\lambda)]| = \epsilon(\lambda)$. We will construct a reduction adversary $\mathcal{B}$ that breaks the mode indistinguishability property of PPS with non-negligible advantage. Description of $\mathcal{B}$ is as follows.

The reduction adversary $\mathcal{B}$ will sample $\tau^* \leftarrow \{0,1\}^\lambda$ and send $(2^\lambda, \tau^*)$ to challenger. $\mathcal{B}$ will receive $\mathsf{pps.crs}, \mathsf{pps.ct}_{\tau^*}$ from PPS challenger. $\mathcal{B}$ will sample $\mathsf{fbe.crs} \leftarrow \mathsf{SSFBE.GSetup}(1^\lambda)$ on its own. $\mathcal{B}$ will set $\mathsf{crs} := (\mathsf{pps.crs}, \mathsf{fbe.crs})$ and send crs to $\mathcal{A}$. Now, $\mathcal{A}$ will make Q-many key generation queries and polynomially many key corruption queries. For the k-th key generation query, if $k < j$, sample $\tau_k \leftarrow \{0,1\}^\lambda$. If τ_k is repeated or $\tau_k = \tau^*$, abort and output a random bit. Otherwise, sample $(\mathsf{fbe.pk}_0, \mathsf{fbe.sk}_0) \leftarrow \mathsf{SSFBE.KGen}$ and $(\mathsf{fbe.pk}_1, \mathsf{fbe.sk}_1) \leftarrow \mathsf{SSFBE.KGen}$. Sample bit $s_k \leftarrow \{0,1\}$. Send $(\tau_k, s_k \oplus 1)$ to PPS challenger and receive $\mathsf{pps.ct}_k$. Set $\mathsf{pk}_k := (\mathsf{fbe.pk}_0, \mathsf{fbe.pk}_1, \tau_k, \mathsf{pps.ct}_k)$, $\mathsf{sk}_k := (s_k, \mathsf{fbe.sk}_{s_k})$ and send pk_k to $\mathcal{A}$. If $k = j$, sample $(\mathsf{fbe.pk}_0, \mathsf{fbe.sk}_0) \leftarrow \mathsf{SSFBE.KGen}$ and $(\mathsf{fbe.pk}_1, \mathsf{fbe.sk}_1) \leftarrow \mathsf{SSFBE.KGen}$. Set $\mathsf{pk}_j := (\mathsf{fbe.pk}_0, \mathsf{fbe.pk}_1, \tau^*, \mathsf{pps.ct}_{\tau^*})$ and send pk_j to $\mathcal{A}$. Notably, do not set sk_j. If $k > j$, sample $\tau_k \leftarrow \{0,1\}^\lambda$. If τ_k is repeated or $\tau_k = \tau^*$, abort and output a random bit. Otherwise, sample $(\mathsf{fbe.pk}_0, \mathsf{fbe.sk}_0) \leftarrow \mathsf{SSFBE.KGen}$ and $(\mathsf{fbe.pk}_1, \mathsf{fbe.sk}_1) \leftarrow \mathsf{SSFBE.KGen}$. Sample bit $s_k \leftarrow \{0,1\}$. Sample $\mathsf{pps.ct}_k \leftarrow \mathsf{PPS.pkEnc}(\tau_k)$. Set $\mathsf{pk}_k := (\mathsf{fbe.pk}_0, \mathsf{fbe.pk}_1, \tau_k, \mathsf{pps.ct}_k)$, $\mathsf{sk}_k := (s_k, \mathsf{fbe.sk}_{s_k})$ and send pk_k to $\mathcal{A}$. For any key corruption queries, respond similarly to $\mathbf{Hyb}_{2,j}^{\mathcal{A}}(1^\lambda) \wedge \mathsf{E}$.

When $\mathcal{A}$ makes a challenge encryption query, let $\mathsf{pk}_1, \ldots, \mathsf{pk}_N$ be the derived public keys. Sample each of them as $(\mathsf{fbe.pk}_{i,0}, \mathsf{fbe.pk}_{i,1}, \tau_i, \mathsf{pps.ct}_i)$ for $i \in [N]$. Compute the set of tags $S = \{\tau_i\}_{i \in [N]}$. Note that $|S| = N$. Send S to PPS challenger and receive $\mathsf{pps.sk}_S$. Then compute $t_i = \mathsf{PPS.Decode}(S, \mathsf{pps.sk}_S, \tau_i, \mathsf{pps.ct}_i)$ for each $i \in [N]$. Then perform encryption for SSFBE similarly to $\mathbf{Hyb}_{2,j}^{\mathcal{A}}(1^\lambda) \wedge \mathsf{E}$. Send $(\mathsf{pps.sk}_S, \mathsf{fbe.ct}_0, \mathsf{fbe.ct}_1)$ to $\mathcal{A}$. Use $\mathcal{A}$ guesses $b' = 0$, send 0 to PPS challenger and 1 otherwise.

Firstly, note that by admissibility of $\mathcal{A}$ and as E takes place, j-th key is not corrupted and thus s_j is no longer used anywhere. As τ^* and $\tau_1, \ldots, \tau_{j-1}, \tau_{j+1}, \ldots, \tau_Q$ are unique, we can set $\tau_j = \tau^*$ as both are from the same distribution. If PPS challenger sample $\mathsf{pps.ct}_j = \mathsf{pps.ct}_{\tau^*}$ using pkEnc, then $\mathcal{B}$ is distributed as $\mathbf{SubHyb}_1^{\mathcal{A}}(1^\lambda)$. Otherwise, it is distributed as $\mathbf{SubHyb}_2^{\mathcal{A}}(1^\lambda)$. Thus, $\mathcal{B}$ can break mode indistinguishability property with advantage $\epsilon(\lambda)$. Thus, the proposition holds. □

Proposition 5.11. $\Pr[1 \leftarrow \mathbf{SubHyb}_2^{\mathcal{A}}(1^\lambda)] = \Pr[1 \leftarrow \mathbf{SubHyb}_3^{\mathcal{A}}(1^\lambda)]$.

Proof. As event E happens, by admissibility of $\mathcal{A}$, the j-th will not be corrupted. It means that the bit s_j of the j-th key generation query is no longer be used in both experiments $\mathbf{SubHyb}_2^{\mathcal{A}}(1^\lambda)$ and $\mathbf{SubHyb}_3^{\mathcal{A}}(1^\lambda)$. So, computing it after the interaction ends with $\mathcal{A}$ will not change the view and thus the advantage of $\mathcal{A}$ in any way. So, proposition holds. □

Proposition 5.12. $\Pr[1 \leftarrow \mathbf{SubHyb}_3^{\mathcal{A}}(1^\lambda)] = \Pr[1 \leftarrow \mathbf{SubHyb}_4^{\mathcal{A}}(1^\lambda)]$.

Proof. As event E happens, by admissibility of $\mathcal{A}$, the j-th will not be corrupted. It means that the bit s_j of the j-th key generation query is no longer be used in both experiments $\mathbf{SubHyb}_3^{\mathcal{A}}(1^\lambda)$ and $\mathbf{SubHyb}_4^{\mathcal{A}}(1^\lambda)$. So, computing it right after the j-th key generation query will not change the view and thus the advantage of $\mathcal{A}$ in any way. So, proposition holds. □

Proposition 5.13. $\Pr[1 \leftarrow \mathbf{SubHyb}_4^{\mathcal{A}}(1^\lambda)] = \Pr[1 \leftarrow \mathbf{Hyb}_{2,j+1}^{\mathcal{A}}(1^\lambda)]$.

Proof. As event E happens, by admissibility of $\mathcal{A}$, the j-th will not be corrupted. It means that the bit s_j of the j-th key generation query is no longer be used in both experiments $\mathbf{SubHyb}_4^{\mathcal{A}}(1^\lambda)$ and $\mathbf{Hyb}_{2,j+1}^{\mathcal{A}}(1^\lambda)$. Moreover, the distribution of s_j in both of these experiment differs as follows.

$$\text{In } \mathbf{SubHyb}_4^{\mathcal{A}}(1^\lambda),\ \mu_j^* \leftarrow \{0,1\} \text{ and } s_j = 1 \oplus \mu_j^*.$$
$$\text{In } \mathbf{Hyb}_{2,j+1}^{\mathcal{A}}(1^\lambda),\ s_j \leftarrow \{0,1\},\ \mu_j^* = 1 \oplus s_j.$$

These are identically distributed. Moreover, as s_j is no longer used, setting or not setting sk_j in the FBE scheme will not affect the view or advantage of $\mathcal{A}$. So, proposition holds. □

Using the above propositions, we prove the claim. □

Claim 5.14. Let E be the event from Lemma 5.7. If PPS satisfies payload hiding, then $|\Pr[1 \leftarrow \mathbf{Hyb}_{2,j}^{\mathcal{A}}(1^\lambda) \wedge \overline{\mathsf{E}}] - \Pr[1 \leftarrow \mathbf{Hyb}_{2,j+1}^{\mathcal{A}}(1^\lambda) \wedge \overline{\mathsf{E}}]| \leq \mathsf{negl}(\lambda)$.

Proof. Here, we can directly apply payload hiding of PPS. Assume for the sake of contradiction that there exists an adversary $\mathcal{A}$ such that $|\Pr[1 \leftarrow \mathbf{Hyb}_{2,j}^{\mathcal{A}}(1^\lambda) \wedge \overline{\mathsf{E}}] - \Pr[1 \leftarrow \mathbf{Hyb}_{2,j+1}^{\mathcal{A}}(1^\lambda) \wedge \overline{\mathsf{E}}]| = \epsilon(\lambda)$. We will construct a reduction adversary $\mathcal{B}$ that breaks the payload hiding property of PPS with non-negligible advantage. Description of $\mathcal{B}$ is as follows.

The reduction adversary $\mathcal{B}$ will sample $\tau^* \leftarrow \{0,1\}^\lambda$, $s^* \leftarrow \{0,1\}$, and send $(2^\lambda, \tau^*, 1 \oplus s^*)$ to challenger. $\mathcal{B}$ will receive $\mathsf{pps.crs}, \mathsf{pps.ct}_{\tau^*}$ from PPS challenger. $\mathcal{B}$ will sample $\mathsf{fbe.crs} \leftarrow \mathsf{SSFBE.GSetup}(1^\lambda)$ on its own. $\mathcal{B}$ will set $\mathsf{crs} := (\mathsf{pps.crs}, \mathsf{fbe.crs})$ and send crs to $\mathcal{A}$. Now, $\mathcal{A}$ will make Q-many key generation queries and polynomially many key corruption queries. For the k-th key generation query, if $k < j$, sample $\tau_k \leftarrow \{0,1\}^\lambda$. If τ_k is repeated or $\tau_k = \tau^*$, abort and output a random bit. Otherwise, sample $(\mathsf{fbe.pk}_0, \mathsf{fbe.sk}_0) \leftarrow \mathsf{SSFBE.KGen}$ and $(\mathsf{fbe.pk}_1, \mathsf{fbe.sk}_1) \leftarrow \mathsf{SSFBE.KGen}$. Sample bit $s_k \leftarrow \{0,1\}$. Send $(\tau_k, s_k \oplus 1)$ to PPS challenger and receive $\mathsf{pps.ct}_k$. Set $\mathsf{pk}_k := (\mathsf{fbe.pk}_0, \mathsf{fbe.pk}_1, \tau_k, \mathsf{pps.ct}_k)$, $\mathsf{sk}_k := (s_k, \mathsf{fbe.sk}_{s_k})$ and send pk_k to $\mathcal{A}$. If $k = j$, sample $(\mathsf{fbe.pk}_0, \mathsf{fbe.sk}_0) \leftarrow \mathsf{SSFBE.KGen}$ and $(\mathsf{fbe.pk}_1, \mathsf{fbe.sk}_1) \leftarrow \mathsf{SSFBE.KGen}$. Set $\mathsf{pk}_j := (\mathsf{fbe.pk}_0, \mathsf{fbe.pk}_1, \tau^*, \mathsf{pps.ct}_{\tau^*})$ and $\mathsf{sk}_j := (s^*, \mathsf{fbe.sk}_{s^*})$ and send pk_j to $\mathcal{A}$. Notably, we can also set sk_j. If $k > j$, sample $\tau_k \leftarrow \{0,1\}^\lambda$. If τ_k is repeated or $\tau_k = \tau^*$, abort and output a random bit. Otherwise, sample $(\mathsf{fbe.pk}_0, \mathsf{fbe.sk}_0) \leftarrow \mathsf{SSFBE.KGen}$ and $(\mathsf{fbe.pk}_1, \mathsf{fbe.sk}_1) \leftarrow \mathsf{SSFBE.KGen}$. Sample bit $s_k \leftarrow \{0,1\}$. Sample $\mathsf{pps.ct}_k \leftarrow \mathsf{PPS.pkEnc}(\tau_k)$. Set $\mathsf{pk}_k := (\mathsf{fbe.pk}_0, \mathsf{fbe.pk}_1, \tau_k, \mathsf{pps.ct}_k)$, $\mathsf{sk}_k := (s_k, \mathsf{fbe.sk}_{s_k})$ and send pk_k to $\mathcal{A}$. For any key corruption queries, respond similarly to $\mathbf{Hyb}_{2,j}^{\mathcal{A}}(1^\lambda) \wedge \overline{\mathsf{E}}$. This means even if the j-th key generation query is corrupted, we can answer that key corruption query.

When $\mathcal{A}$ makes a challenge encryption query, let $\mathsf{pk}_1, \ldots, \mathsf{pk}_N$ be the derived public keys. Sample each of them as $(\mathsf{fbe.pk}_{i,0}, \mathsf{fbe.pk}_{i,1}, \tau_i, \mathsf{pps.ct}_i)$ for $i \in [N]$. Compute the set of tags $S = \{\tau_i\}_{i\in[N]}$. Note that $|S| = N$. Send S to PPS challenger and receive $\mathsf{pps.sk}_S$. Then compute $t_i = \mathsf{PPS.Decode}(S, \mathsf{pps.sk}_S, \tau_i, \mathsf{pps.ct}_i)$ for each $i \in [N]$. Then perform encryption for SSFBE similarly to $\mathbf{Hyb}^{\mathcal{A}}_{2,j}(1^\lambda) \wedge \overline{\mathsf{E}}$. Send $(\mathsf{pps.sk}_S, \mathsf{fbe.ct}_0, \mathsf{fbe.ct}_1)$ to $\mathcal{A}$. Use $\mathcal{A}$ guesses $b' = 0$, send 0 to PPS challenger and 1 otherwise.

Firstly, note that by admissibility of $\mathcal{A}$ and as $\overline{\mathsf{E}}$ takes place $\tau_j \notin S$. As τ^* and $\tau_1, \ldots, \tau_{j-1}, \tau_{j+1}, \ldots, \tau_Q$ are unique, we can set $\tau_j = \tau^*$ as both are from the same distribution. If PPS challenger sample $\mathsf{pps.ct}_j = \mathsf{pps.ct}_{\tau^*}$ using pkEnc, then $\mathcal{B}$ is distributed as $\mathbf{Hyb}^{\mathcal{A}}_{2,j}(1^\lambda)$. Otherwise, it is distributed as $\mathbf{Hyb}^{\mathcal{A}}_{2,j+1}(1^\lambda)$. Thus, $\mathcal{B}$ can break payload hiding property with advantage $\epsilon(\lambda)$. Thus, the claim holds. □

Using the above claims, we prove the lemma. □

Lemma 5.15. $\Pr[1 \leftarrow \mathbf{Hyb}^{\mathcal{A}}_{2,Q+1}(1^\lambda)] = \Pr[1 \leftarrow \mathbf{Hyb}^{\mathcal{A}}_{3}(1^\lambda)]$.

Proof. This follows as $\mathbf{Hyb}^{\mathcal{A}}_{3}(1^\lambda)$ is same as $\mathbf{Hyb}^{\mathcal{A}}_{2,Q+1}(1^\lambda)$ and it is explicitly written out for convenience. □

Lemma 5.16. If SSFBE satisfies semi-static security (Definition 5.2), then for any PPT admissible adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that $|\Pr[1 \leftarrow \mathbf{Hyb}^{\mathcal{A}}_{3}(1^\lambda)] - \Pr[1 \leftarrow \mathbf{Hyb}^{\mathcal{A}}_{4}(1^\lambda)]| \leq \mathsf{negl}(\lambda)$.

Proof. Assume for contradiction that there exists a stateful PPT adversary $\mathcal{A}$ such that $|\Pr[1 \leftarrow \mathbf{Hyb}^{\mathcal{A}}_{3}(1^\lambda)] - \Pr[1 \leftarrow \mathbf{Hyb}^{\mathcal{A}}_{4}(1^\lambda)]| = \epsilon(\lambda)$ for some non-negligible function $\epsilon(\cdot)$. We will use $\mathcal{A}$ to construction a reduction adversary $\mathcal{B}$ which will break the semi-static security property of SSFBE with non-negligible probability. Description of $\mathcal{B}$ is as follows.

$\mathcal{B}$ receives $\mathsf{fbe.crs}$ from the SSFBE challenger. It will sample $\mathsf{pps.crs} \leftarrow \mathsf{PPS.GSetup}(1^\lambda, 2^\lambda)$ and $\mathsf{pps.msk} \leftarrow \mathsf{PPS.MSKGen}$ on its own and send $\mathsf{crs} := (\mathsf{pps.crs}, \mathsf{fbe.crs})$ to $\mathcal{A}$. Whenever $\mathcal{A}$ queries for key generation, sample $\tau_j \leftarrow \{0,1\}^\lambda$. If τ_j is repeated abort and output and random bit. Otherwise, sample $s_j \leftarrow \{0,1\}$, $\mathsf{pps.ct}_j \leftarrow \mathsf{PPS.skEnc}(\mathsf{pps.msk}, \tau_j, 1 \oplus s_j)$. Then, sample $(\mathsf{fbe.pk}_{s_j}, \mathsf{fbe.sk}_{s_j}) \leftarrow \mathsf{SSFBE.KGen}$. Query the SSFBE challenger for a public key and set it as $\mathsf{fbe.pk}_{1\oplus s_j}$. Set $\mathsf{sk}_j = (s_j, \mathsf{fbe.sk}_{s_j})$ and send $\mathsf{pk}_j := (\mathsf{fbe.pk}_0, \mathsf{fbe.pk}_1, \tau_j, \mathsf{pps.ct}_j)$ to $\mathcal{A}$. Whenever $\mathcal{A}$ makes a key corruption query, send the corresponding secret key.

When $\mathcal{A}$ makes a challenge encryption query, given $\mathsf{pk}_1, \ldots, \mathsf{pk}_N$, parse each of them as $(\mathsf{fbe.pk}_{i,0}, \mathsf{fbe.pk}_{i,1}, \tau_i, \mathsf{pps.ct}_i)$ for each $i \in [N]$. Compute the set of tags $S = \{\tau_i\}_{i\in[N]}$. Sample $\mathsf{pps.sk}_S \leftarrow \mathsf{PPS.KGen}(\mathsf{pps.msk}, S)$. Then, compute $t_i = \mathsf{PPS.Decode}(S, \mathsf{pps.sk}_S, i, \mathsf{pps.ct}_i)$ and send the handles corresponding to $(\mathsf{fbe.pk}_{i,t_i})_{i\in[N]}$ to SSFBE challenger to receive a ciphertext which $\mathcal{B}$ sets as $\mathsf{fbe.ct}_0$. $\mathcal{B}$ will compute $\mathsf{fbe.ct}_1$ for message 0 locally on its own similarly to $\mathbf{Hyb}^{\mathcal{A}}_{4}(1^\lambda)$. $\mathcal{B}$ will send $\mathsf{ct} := (\mathsf{pps.sk}_S, \mathsf{fbe.ct}_0, \mathsf{fbe.ct}_1)$ to $\mathcal{A}$. $\mathcal{B}$ will pass $\mathcal{A}$'s guess b' as its own.

Note that as each $\mathsf{pps.ct}_j$ is sampled using $\mathsf{PPS.skEnc}$, we have that $t_i = 1 \oplus s_i$ in the challenge query phase. Thus, $\mathsf{fbe.pk}_{i,t_i} = \mathsf{fbe.pk}_{i,1\oplus s_i}$ for each $i \in [N]$. These are the keys we got from the SSFBE challenger and none of these are corrupted as we send the s_j-th instance for key corruption (which $\mathcal{B}$ samples on its own). Thus, $\mathcal{B}$ is a valid adversary against SSFBE. If SSFBE challenger encrypted bit 0, then $\mathcal{B}$ behaves like $\mathbf{Hyb}^{\mathcal{A}}_{3}(1^\lambda)$. Otherwise, $\mathcal{B}$ behaves like $\mathbf{Hyb}^{\mathcal{A}}_{4}(1^\lambda)$. Thus, $\mathcal{B}$ breaks the semi-static security property of SSFBE with non-negligible probability $\epsilon(\lambda)$. Thus, lemma holds. □

Lemma 5.17. If PPS satisfies mode indistinguishability and payload hiding properties (Definition 4.1), then for any PPT admissible adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that $|\Pr[1 \leftarrow \mathbf{Hyb}_4^{\mathcal{A}}(1^\lambda)] - \Pr[1 \leftarrow \mathbf{Hyb}_5^{\mathcal{A}}(1^\lambda)]| \leq \mathsf{negl}(\lambda)$.

Proof. This is similar to proof of Lemma 5.7 except in the opposite direction. □

Lemma 5.18. If PPS satisfies mode indistinguishability and payload hiding properties (Definition 4.1), then for any PPT admissible adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that $|\Pr[1 \leftarrow \mathbf{Hyb}_5^{\mathcal{A}}(1^\lambda)] - \Pr[1 \leftarrow \mathbf{Hyb}_6^{\mathcal{A}}(1^\lambda)]| \leq \mathsf{negl}(\lambda)$.

Proof. This is similar to proof of Lemma 5.7 except in the now we encrypt s instead of $1 \oplus s$ inside PPS.skEnc. □

Lemma 5.19. If SSFBE satisfies semi-static security (Definition 5.2), then for any PPT admissible adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that $|\Pr[1 \leftarrow \mathbf{Hyb}_6^{\mathcal{A}}(1^\lambda)] - \Pr[1 \leftarrow \mathbf{Hyb}_7^{\mathcal{A}}(1^\lambda)]| \leq \mathsf{negl}(\lambda)$.

Proof. This is similar to proof of Lemma 5.16 except now each $t_i = s_i$. In a bit more detail, we will still request from the challenger public keys for $\mathsf{fbe.pk}_{1 \oplus s_j}$ and give out $(s_j, \mathsf{fbe.sk}_{s_j})$ for key corruption. Thus, we can argue security for $\mathsf{fbe.ct}_1$. □

Lemma 5.20. If PPS satisfies mode indistinguishability and payload hiding properties (Definition 4.1), then for any PPT admissible adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that $|\Pr[1 \leftarrow \mathbf{Hyb}_7^{\mathcal{A}}(1^\lambda)] - \Pr[1 \leftarrow \mathbf{Hyb}_8^{\mathcal{A}}(1^\lambda)]| \leq \mathsf{negl}(\lambda)$.

Proof. There are two changes made. Firstly, we sample each $\mathsf{pps.ct}_\tau$ during key generation phase using $\mathsf{PPS.pkEnc}(\tau)$ as opposed to $\mathsf{PPS.skEnc}(\mathsf{pps.msk}, \tau, s)$ as in $\mathbf{Hyb}_7^{\mathcal{A}}(1^\lambda)$. This change is indistinguishable to $\mathcal{A}$ and can be argued similarly to proof of Lemma 5.7.

Secondly, we stop checking if each $\tau_1, \ldots, \tau_Q$ are unique where τ_j is generated during the j-th key generation query. Moreover, we also sample pps.msk during challenge encryption phase. This abort condition will only be triggered with negligible probability and indistinguishability can be argued similarly to Lemma 5.5. □

Using the above lemmas, we prove the theorem. □

Corollary 5.21. Assuming the hardness of decomposed LWE assumption with sub-exponential modulus-to-noise ratio, there exists an adaptively secure flexible broadcast encryption scheme in the standard model where each parameter grows with $\mathsf{poly}(\lambda)$.

Proof. Goyal and Yadugiri [GY26a] and Abram, Malavolta, and Roy [AMR26] recently constructed flexible broadcast encryption schemes from decomposed LWE assumption with semi-static security (Definition 5.2) and optimal parameters where every parameter only grows with $\mathsf{poly}(\lambda)$ including the ciphertext. Both works then apply the Gentry-Waters compiler [GW09] with a random oracle to obtain adaptive security. The semi-statically secure scheme of [GY26a] relies on a random oracle whereas that of [AMR26] is in the standard model. Our compiler (Construction 5.3, Theorem 5.4) compiles the latter into an adaptively secure FBE scheme (Definition 5.1) in the standard model with optimal parameters. □

Applying Construction 5.3 to the semi-statically secure scheme of [GY26a] instead gives an adaptively secure FBE scheme with optimal parameters in the random oracle model, which [GY26a] already achieve. We believe that it is possible to also remove the random oracle from [GY26a]'s semi-statically secure scheme while preserving compactness, but we do not pursue this here.

Note that by Corollary 5.21, we have an adaptively secure FBE in the standard model with optimal parameters from decomposed LWE. Moreover, it is easy to see that by [AMR26], this FBE scheme has pseudorandom and back sampleable public keys as defined by [GY26a]. Similarly, by Remark 4.7 we have $\mathsf{pps.ct}_\tau$ in each FBE public key to be uniformly random string. Then, using the techniques in [GY26a], we can place these public keys in a random oracle to get an IBBE scheme with a *single* random oracle. [GY26a] required 2 random oracles for their construction of IBBE.

5.2 Distributed BE and Alternatives

Applications to Distributed Broadcast Encryption. In distributed broadcast encryption (DBE) [WQZD10, BZ14], GSetup takes the number of users N as input and each user $i \in [N]$ generates its own key pair. Hsieh, Waters, and Wu [HWW25, Appendix A] compile any semi-statically secure DBE scheme into an adaptively secure one in the standard model. Their compiler incurs an at least linear blow up in the crs. If we instead use a compiler, similar to Construction 5.3, each user i can publish its own PPS ciphertext $\mathsf{pps.ct}_i \leftarrow \mathsf{PPS.pkEnc}(i)$ along with two public keys of a semi-statically secure DBE scheme for $2N$ users, one for each slot $(i, b) \in [N] \times \{0, 1\}$, and keep a random bit s_i along with the secret key for slot (i, s_i). This compiler only adds $|\mathsf{pps.crs}| = \mathsf{poly}(\lambda, \log N)$ to the crs and a single PPS ciphertext, which is a uniformly random string, to each public key. Adaptive security follows the same hybrids as the proof of Theorem 5.4 with further simplifications.

We remark that Theorem 5.4 along with the semi-statically secure FBE scheme of [AMR26] already yields an adaptively secure DBE scheme with optimal parameters in the standard model as FBE implies DBE. However, the above route might avoid the expensive tree encodings of [AMR26] and provide a more direct route to adaptively secure DBE via matrix commitments [WW25, GY26c] with a potentially long crs. Our PPS scheme also improves the adaptively secure DBE schemes of [GY26c] that use PPS to remove random oracles.

Corollary 5.22. Assuming the hardness of $2m^2$-decomposed LWE with sub-exponential modulus-to-noise ratio, there exists an adaptively secure DBE scheme for N users in the standard model with transparent setup where the public key, secret key, and ciphertext sizes are $\mathsf{poly}(\lambda, \log N)$ and $|\mathsf{crs}| = N \cdot \mathsf{poly}(\lambda, \log N)$.

Proof. This follows from [GY26c, Construction 7.1] and Corollary 4.30. Notably, this eliminates the need for 1-key LWE for linear sized crs in DBE from [GY26c]. □

The above corollary also matches the DBE result of [AMR26]. However, as mentioned, it avoids the expensive tree encodings and gives an alternate construction from matrix commitments [WW25, GY26c]. This way, we can stay under the bar of what is required for FBE.

6 Application II: Projective PRGs and Succinct Computational Secret Sharing

In this section, we give our second application of PPS: projective PRGs (PPRGs) [ABI+23] and succinct computational secret sharing (SCSS). A PPRG expands a short seed into a long pseudorandom string such that a short projected key for a set S reveals exactly the outputs at the indices in S. Hsieh, Waters, and Wu [HWW25] consider publicly sampleable PPRGs (PSPRGs) where a projected key can also be sampled without the seed.

From PPS to PSPRG. The connection is direct. The public parameters consist of a PPS crs and a secret-key PPS ciphertext $\mathsf{pps.ct}_i \leftarrow \mathsf{PPS.skEnc}(\mathsf{pps.msk}, i, \mu_i)$ of a random bit μ_i for each index i, the seed is $\mathsf{pps.msk}$, the projected key for S is $\mathsf{pps.sk}_S \leftarrow \mathsf{PPS.KGen}(\mathsf{pps.msk}, S)$, and the i-th output is $\mathsf{PPS.Decode}(S, \mathsf{pps.sk}_S, i, \mathsf{pps.ct}_i)$. Payload hiding guarantees that the outputs outside S are pseudorandom given $\mathsf{pps.sk}_S$. A projected key can be sampled publicly as a PPS secret key under a fresh master secret key, which is indistinguishable from an honest one by mode indistinguishability and payload hiding.

From PSPRG to PPS? The converse does not appear to be immediate as the PSPRG definition of [HWW25, Definition 4.1] does not provide all the guarantees of PPS. To begin with, Setup of a PSPRG is private-coin and does not allow msk to be sampled independently of pp. Even if it did, consider the natural candidate with $\mathcal{M} = \{0,1\}$ where $\mathsf{pps.msk} := (\mathsf{msk}, \mathsf{sk}_{[\ell_{\mathsf{out}}]})$, $\mathsf{PPS.skEnc}(\mathsf{pps.msk}, i, \mu) := \mu \oplus \mathsf{Eval}([\ell_{\mathsf{out}}], \mathsf{sk}_{[\ell_{\mathsf{out}}]}, i)$, $\mathsf{PPS.pkEnc}(i)$ outputs a random bit, and $\mathsf{pps.sk}_S := \mathsf{Project}(\mathsf{msk}, S)$. Correctness and mode indistinguishability are immediate. The difficulty is payload hiding. Let P be the set of indices for which $\mathcal{A}$ requests skEnc ciphertexts with messages of its choice. From these, $\mathcal{A}$ learns $\mathsf{Eval}([\ell_{\mathsf{out}}], \mathsf{sk}_{[\ell_{\mathsf{out}}]}, j)$ for every $j \in P$, and it may then choose a set S with $S \cap P \neq \emptyset$. Adaptive pseudorandomness of PSPRG only guarantees security when $S \cap P = \emptyset$, so a reduction cannot obtain sk_S. In other words, PPS lets a reduction program the message at every index other than the target while the target stays hidden, whereas a PSPRG only hides the outputs outside the projected set.

Roadmap. In Section 6.1, we define PSPRGs following [HWW25] and construct them from PPS. In Section 6.2, we define a variant of PPS which we call robust PPS, where payload hiding holds against any number of secret keys and mode indistinguishability is dropped, and we use it to construct SCSS for graph access structures where the vertices join one at a time. In Section 6.3, we construct robust PPS from batched IBE and from ABE with succinct keys. In Section A, we show that PPS gives a robust PPS against a bounded number of secret keys, which yields SCSS for graphs with a bounded number of corrupted vertices and no public share from PPS alone.

6.1 Publicly Sampleable Projective PRGs from PPS

We define publicly sampleable projective PRGs following [HWW25].

Syntax. A publicly sampleable projective pseudorandom generator scheme (PSPRG) consists of the following polynomial time algorithms.

$\mathsf{Setup}(1^\lambda, 1^{\ell_{\mathsf{out}}}) \to (\mathsf{pp}, \mathsf{msk})$**.** The randomized setup algorithm takes as input security parameter λ, output length ℓ_{out}, and outputs public parameters pp and master secret key msk. The following algorithms take pp implicitly.

$\mathsf{Samp}(S) \to \mathsf{sk}_S$**.** The randomized public sampling algorithm takes as input set $S \subseteq [\ell_{\mathsf{out}}]$ and outputs secret key sk_S.

$\mathsf{Project}(\mathsf{msk}, S) \to \mathsf{sk}_S$**.** The possibly randomized projection algorithm takes as input master secret key msk, set $S \subseteq [\ell_{\mathsf{out}}]$, and outputs secret key sk_S.

$\mathsf{Eval}(S, \mathsf{sk}_S, i) \to y_i$. The deterministic evaluation algorithm takes as input set $S \subseteq [\ell_{\mathsf{out}}]$, secret key sk_S, index $i \in S$, and outputs a $y_i \in \{0,1\}$.

Definition 6.1 (PSPRG)**.** A PSPRG scheme $(\mathsf{Setup}, \mathsf{Samp}, \mathsf{Project}, \mathsf{Eval})$ is said to be a publicly sampleable projective PRG scheme if it satisfies the following properties.

Correctness. For any $\lambda \in \mathbb{N}$, $\ell_{\mathsf{out}} = \ell_{\mathsf{out}}(\lambda)$, $S \subseteq [\ell_{\mathsf{out}}]$, and $i \in S$,

$$\Pr\left[\mathsf{Eval}([\ell_{\mathsf{out}}], \mathsf{sk}_{[\ell_{\mathsf{out}}]}, i) = \mathsf{Eval}(S, \mathsf{sk}_S, i) \quad : \quad \begin{array}{l}(\mathsf{pp}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda, 1^{\ell_{\mathsf{out}}}),\\ \mathsf{sk}_{[\ell_{\mathsf{out}}]} \leftarrow \mathsf{Project}(\mathsf{msk}, [\ell_{\mathsf{out}}]),\\ \mathsf{sk}_S \leftarrow \mathsf{Project}(\mathsf{msk}, S)\end{array}\right] = 1$$

Compactness. There exists a polynomial $\mathsf{poly}(\cdot,\cdot)$ such that for any $\lambda \in \mathbb{N}$, $\ell_{\mathsf{out}} = \mathsf{poly}(\lambda)$, and $S \subseteq [\ell_{\mathsf{out}}]$, $|\mathsf{sk}_S| = \mathsf{poly}(\lambda, \log \ell_{\mathsf{out}})$.

Sampling Indistinguishability. For any admissible PPT adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that $\forall\ \lambda \in \mathbb{N}$,

$$\left|\Pr\left[b \leftarrow \mathcal{A}(\mathsf{sk}_S) \quad : \quad \begin{array}{l}1^{\ell_{\mathsf{out}}} \leftarrow \mathcal{A}(1^\lambda),\\ (\mathsf{pp}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda, 1^{\ell_{\mathsf{out}}}),\\ S \leftarrow \mathcal{A}(\mathsf{pp}), b \leftarrow \{0,1\},\\ \text{if } b = 0, \mathsf{sk}_S \leftarrow \mathsf{Samp}(S),\\ \text{if } b = 1, \mathsf{sk}_S \leftarrow \mathsf{Project}(\mathsf{msk}, S)\end{array}\right] - \frac{1}{2}\right| \leq \mathsf{negl}(\lambda)$$

A PPT machine $\mathcal{A}$ is said to be admissible if $S \subseteq [\ell_{\mathsf{out}}]$.

Adaptive Pseudorandomness. For any admissible PPT adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for any $\lambda \in \mathbb{N}$,

$$\left|\Pr\left[b \leftarrow \mathcal{A}^{\mathcal{O}_b(\cdot,\cdot)}(\mathsf{pp}) \quad : \quad \begin{array}{l}\mathcal{I} := \emptyset, \mathcal{T} := \emptyset, 1^{\ell_{\mathsf{out}}} \leftarrow \mathcal{A}(1^\lambda),\\ (\mathsf{pp}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda, 1^{\ell_{\mathsf{out}}}),\\ \mathsf{sk}_{[\ell_{\mathsf{out}}]} \leftarrow \mathsf{Project}(\mathsf{msk}, [\ell_{\mathsf{out}}]),\\ \mathbf{y} \leftarrow \{0,1\}^{\ell_{\mathsf{out}}}, b \leftarrow \{0,1\}\end{array}\right] - \frac{1}{2}\right| \leq \mathsf{negl}(\lambda)$$

where $\mathcal{O}_0, \mathcal{O}_1$ are stateful oracles that update $\mathcal{I}, \mathcal{T}$ and respond to the following queries.

- $(\mathsf{Project}, S)$**.** Given $S \subseteq [\ell_{\mathsf{out}}]$, both $\mathcal{O}_0$ and $\mathcal{O}_1$ reply with $\mathsf{sk}_S \leftarrow \mathsf{Project}(\mathsf{msk}, S)$. After this, both oracles update $\mathcal{T} := \mathcal{T} \cup S$.
- (Eval, i)**.** Given $i \in [\ell_{\mathsf{out}}]$, $\mathcal{O}_0$ replies with $\mathsf{Eval}([\ell_{\mathsf{out}}], \mathsf{sk}_{[\ell_{\mathsf{out}}]}, i)$. $\mathcal{O}_1$ on the other hand replies with y_i, the i-th bit of $\mathbf{y} \in \{0,1\}^{\ell_{\mathsf{out}}}$. After this, both oracles update $\mathcal{I} := \mathcal{I} \cup \{i\}$.

A PPT machine $\mathcal{A}$ is said to be admissible if it makes at most one query of the form $(\mathsf{Project}, \cdot)$ and $\mathcal{I} \cap \mathcal{T} = \emptyset$ at the end of the experiment.

We now show how to use PPS to instantiate a PSPRG scheme.

Construction 6.2. We construct a PSPRG scheme (Definition 6.1) using a PPS scheme for $\mathcal{M} = \{0,1\}$ (Definition 4.1) as follows.

$\mathsf{Setup}(1^\lambda, 1^{\ell_{\mathsf{out}}})$**.** Sample $\mathsf{pps.crs} \leftarrow \mathsf{PPS.GSetup}(1^\lambda, \ell_{\mathsf{out}})$ and $\mathsf{pps.msk} \leftarrow \mathsf{PPS.MSKGen}$. For each $i \in [\ell_{\mathsf{out}}]$, sample $\mu_i \leftarrow \{0,1\}$ and $\mathsf{pps.ct}_i \leftarrow \mathsf{PPS.skEnc}(\mathsf{pps.msk}, i, \mu_i)$.

Output $\mathsf{pp} := (\mathsf{pps.crs}, (\mathsf{pps.ct}_i)_{i\in[\ell_{\mathsf{out}}]})$ and $\mathsf{msk} := \mathsf{pps.msk}$.

$\mathsf{Samp}(S)$**.** Sample $\mathsf{pps.msk}' \leftarrow \mathsf{PPS.MSKGen}$ and $\mathsf{pps.sk}'_S \leftarrow \mathsf{PPS.KGen}(\mathsf{pps.msk}', S)$.

Output $\mathsf{sk}_S := \mathsf{pps.sk}'_S$.

$\mathsf{Project}(\mathsf{msk}, S)$**.** Parse msk as $\mathsf{pps.msk}$. Sample $\mathsf{pps.sk}_S \leftarrow \mathsf{PPS.KGen}(\mathsf{pps.msk}, S)$.

Output $\mathsf{sk}_S := \mathsf{pps.sk}_S$.

$\mathsf{Eval}(S, \mathsf{sk}_S, i)$**.** If $i \notin S$, output $\bot$. Otherwise, parse sk_S as $\mathsf{pps.sk}_S$.

Output $\mathsf{PPS.Decode}(S, \mathsf{pps.sk}_S, i, \mathsf{pps.ct}_i)$.

Correctness. This follows from correctness of PPS. Specifically, for any $S \subseteq [\ell_{\mathsf{out}}], i \in S$, $\mathsf{sk}_{[\ell_{\mathsf{out}}]} := \mathsf{PPS.KGen}(\mathsf{pps.msk}, [\ell_{\mathsf{out}}])$ and $\mathsf{sk}_S := \mathsf{PPS.KGen}(\mathsf{pps.msk}, S)$. As $i \in S$, by correctness of PPS, $\mathsf{pps.ct}_i := \mathsf{PPS.skEnc}(\mathsf{pps.msk}, i, \mu_i)$, $\mathsf{Eval}(S, \mathsf{sk}_S, i) := \mathsf{PPS.Decode}(S, \mathsf{pps.sk}_S, i, \mathsf{pps.ct}_i) = \mu_i$ for a randomly chosen μ_i during Setup. Similarly, $\mathsf{Eval}([\ell_{\mathsf{out}}], \mathsf{sk}_{[\ell_{\mathsf{out}}]}, i) := \mathsf{PPS.Decode}(S, \mathsf{pps.sk}_{[\ell_{\mathsf{out}}]}, i, \mathsf{pps.ct}_i) = \mu_i$ for the same μ_i as $i \in [\ell_{\mathsf{out}}]$ and by correctness of PPS.

Compactness. This follows from compactness of PPS as $\mathsf{sk}_S := \mathsf{pps.sk}_S$.

Sampling Indistinguishability Analysis

Theorem 6.3 (Sampling Indistinguishability)**.** If PPS satisfies mode indistinguishability and payload hiding properties (Definition 4.1), then Construction 6.2 satisfies sampling indistinguishability property (Definition 6.1).

Proof. We prove this using the following hybrids.

$\mathbf{Hyb}_j^{\mathcal{A}}(1^\lambda)$ **for** $j \in [0, \ell_{\mathsf{out}}]$**.** This is the sampling indistinguishability experiment from Definition 6.1 with $b = 1$ except that we will sample $\mathsf{pps.ct}_i \leftarrow \mathsf{PPS.pkEnc}(i)$ for any $i \le j$.

Setup Phase. $\mathcal{A}$ sends $1^{\ell_{\mathsf{out}}}$. Sample $\mathsf{pps.crs} \leftarrow \mathsf{PPS.GSetup}(1^\lambda, \ell_{\mathsf{out}})$. Sample $\mathsf{pps.msk} \leftarrow \mathsf{PPS.MSKGen}$. Sample $\mu_i \leftarrow \{0,1\}$ for each $i \in [\ell_{\mathsf{out}}]$. For each $i \in [\ell_{\mathsf{out}}]$, $i \le j$, sample $\mathsf{pps.ct}_i \leftarrow \mathsf{PPS.pkEnc}(i)$. Otherwise, sample $\mathsf{pps.ct}_i \leftarrow \mathsf{PPS.skEnc}(\mathsf{pps.msk}, i, \mu_i)$.

Set $\mathsf{pp} := (\mathsf{pps.crs}, (\mathsf{pps.ct}_i)_{i\in[\ell_{\mathsf{out}}]})$, $\mathsf{msk} := \mathsf{pps.msk}$. Send pp to $\mathcal{A}$.

Challenge Phase. $\mathcal{A}$ sends $S \subseteq [\ell_{\mathsf{out}}]$. Sample $\mathsf{pps.sk}_S \leftarrow \mathsf{PPS.KGen}(\mathsf{pps.msk}, S)$. Set $\mathsf{sk}_S := \mathsf{pps.sk}_S$. Send sk_S to $\mathcal{A}$.

Guess Phase. $\mathcal{A}$ outputs its guess b'. Output b'.

$\mathbf{Hyb}^{\mathcal{A}}_{\ell_{\mathsf{out}}+1}(1^\lambda)$**.** This is the same as $\mathbf{Hyb}^{\mathcal{A}}_{\ell_{\mathsf{out}}}(1^\lambda)$ where each $\mathsf{pps.ct}_i \leftarrow \mathsf{PPS.pkEnc}(i)$ for $i \in [\ell_{\mathsf{out}}]$. However, now we will additionally sample $\mathsf{pps.msk}' \leftarrow \mathsf{PPS.MSKGen}$ during setup phase and generate $\mathsf{pps.sk}'_S \leftarrow \mathsf{PPS.KGen}(\mathsf{pps.msk}', S)$ during challenge phase. We will then set $\mathsf{sk}_S := \mathsf{pps.sk}'_S$ and send sk_S to $\mathcal{A}$.

$\mathbf{Hyb}^{\mathcal{A}}_{\ell_{\mathsf{out}}+1+j}(1^\lambda)$ **for** $j \in [\ell_{\mathsf{out}}]$**.** This is the same as $\mathbf{Hyb}^{\mathcal{A}}_{\ell_{\mathsf{out}}+1}(1^\lambda)$ except that we will now sample $\mathsf{pps.ct}_i \leftarrow \mathsf{PPS.skEnc}(\mathsf{pps.msk}, i, \mu_i)$ for a random $\mu_i \leftarrow \{0,1\}$ for any $i \le j$. Note that $\mathbf{Hyb}^{\mathcal{A}}_{2\cdot\ell_{\mathsf{out}}+1}(1^\lambda)$ is exactly the sampling indistinguishability experiment with $b = 0$.

Lemma 6.4. If PPS satisfies mode indistinguishability and payload hiding properties (Definition 4.1) then for any $j \in [\ell_{\mathsf{out}}]$, admissible PPT adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for any $\lambda \in \mathbb{N}$, $|\Pr[1 \leftarrow \mathbf{Hyb}^{\mathcal{A}}_{j-1}(1^\lambda)] - \Pr[1 \leftarrow \mathbf{Hyb}^{\mathcal{A}}_{j}(1^\lambda)]| \le \mathsf{negl}(\lambda)$.

Proof. Note that the only difference between these two hybrid experiments is that we are changing the distribution of $\mathsf{pps.ct}_j$. In $\mathbf{Hyb}^{\mathcal{A}}_{j-1}(1^\lambda)$, we sample it using $\mathsf{PPS.skEnc}(\mathsf{pps.msk}, j, \mu_j)$ and in $\mathbf{Hyb}^{\mathcal{A}}_{j}(1^\lambda)$, we sample it using $\mathsf{PPS.pkEnc}(j)$. We will prove this lemma using piecewise guessing argument [JKK+17, KW20]. We will guess one bit corresponding to whether the $j \in S$ or not. That is, let E denote the event that $j \in S$. Let $\overline{\mathsf{E}}$ denote the complement of this event. Then,

$$\Pr[1 \leftarrow \mathbf{Hyb}^{\mathcal{A}}_{j-1}(1^\lambda)] = \Pr[1 \leftarrow \mathbf{Hyb}^{\mathcal{A}}_{j-1}(1^\lambda) \wedge \mathsf{E}] + \Pr[1 \leftarrow \mathbf{Hyb}^{\mathcal{A}}_{j-1}(1^\lambda) \wedge \overline{\mathsf{E}}]$$
$$\Pr[1 \leftarrow \mathbf{Hyb}^{\mathcal{A}}_{j}(1^\lambda)] = \Pr[1 \leftarrow \mathbf{Hyb}^{\mathcal{A}}_{j}(1^\lambda) \wedge \mathsf{E}] + \Pr[1 \leftarrow \mathbf{Hyb}^{\mathcal{A}}_{j}(1^\lambda) \wedge \overline{\mathsf{E}}]$$

Hence, it enough to show that $|\Pr[1 \leftarrow \mathbf{Hyb}^{\mathcal{A}}_{j-1}(1^\lambda) \wedge \mathsf{E}] - \Pr[1 \leftarrow \mathbf{Hyb}^{\mathcal{A}}_{j}(1^\lambda) \wedge \mathsf{E}]| \le \mathsf{negl}(\lambda)$ and $|\Pr[1 \leftarrow \mathbf{Hyb}^{\mathcal{A}}_{j-1}(1^\lambda) \wedge \overline{\mathsf{E}}] - \Pr[1 \leftarrow \mathbf{Hyb}^{\mathcal{A}}_{j}(1^\lambda) \wedge \overline{\mathsf{E}}]| \le \mathsf{negl}(\lambda)$ to prove this lemma.

Claim 6.5. Let E be the event from Lemma 6.4. If PPS satisfies mode indistinguishability, then $|\Pr[1 \leftarrow \mathbf{Hyb}^{\mathcal{A}}_{j-1}(1^\lambda) \wedge \mathsf{E}] - \Pr[1 \leftarrow \mathbf{Hyb}^{\mathcal{A}}_{j}(1^\lambda) \wedge \mathsf{E}]| \le \mathsf{negl}(\lambda)$.

Proof. Assume for the sake of contradiction that there exists an adversary $\mathcal{A}$ such that $|\Pr[1 \leftarrow \mathbf{Hyb}^{\mathcal{A}}_{j-1}(1^\lambda) \wedge \mathsf{E}] - \Pr[1 \leftarrow \mathbf{Hyb}^{\mathcal{A}}_{j}(1^\lambda) \wedge \mathsf{E}]| = \epsilon(\lambda)$ for a non-negligible function $\epsilon(\cdot)$. We will use $\mathcal{A}$ to construct a reduction adversary $\mathcal{B}$ that will break the mode indistinguishability property of PPS with non-negligible advantage. Description of $\mathcal{B}$ is as follows.

$\mathcal{B}$ receives $1^{\ell_{\mathsf{out}}}$ from $\mathcal{A}$. It will send (ℓ_{out}, j) to PPS mode indistinguishability challenger. $\mathcal{B}$ will receive $(\mathsf{pps.crs}, \mathsf{pps.ct}_j)$. $\mathcal{B}$ will then sample $\mu_i \leftarrow \{0,1\}$ for each $i > j$ and send (i, μ_i) to PPS challenger to receive $\mathsf{pps.ct}_i$. $\mathcal{B}$ will then sample $\mathsf{pps.ct}_i \leftarrow \mathsf{PPS.pkEnc}(i)$ for each $i < j$. It will set $\mathsf{pp} := (\mathsf{pps.crs}, (\mathsf{pps.ct}_i)_{i\in[\ell_{\mathsf{out}}]})$ and send pp to $\mathcal{A}$. $\mathcal{A}$ will then send $S \subseteq [\ell_{\mathsf{out}}]$ such that $j \in S$. $\mathcal{B}$ will pass this S to PPS challenger and receive $\mathsf{pps.sk}_S$. $\mathcal{B}$ will set $\mathsf{sk}_S := \mathsf{pps.sk}_S$ and send sk_S to $\mathcal{A}$. $\mathcal{B}$ will output $\mathcal{A}$'s guess b' as its own.

Firstly, note that $\mathcal{B}$ is a valid adversary against the mode indistinguishability game. If PPS challenger samples $\mathsf{pps.ct}_j$ using $\mathsf{PPS.skEnc}$, then $\mathcal{B}$ behaves like $\mathbf{Hyb}^{\mathcal{A}}_{j-1}(1^\lambda) \wedge \mathsf{E}$ and if PPS challenger samples $\mathsf{pps.ct}_j$ using $\mathsf{PPS.pkEnc}$, then $\mathcal{B}$ behaves like $\mathbf{Hyb}^{\mathcal{A}}_{j}(1^\lambda) \wedge \mathsf{E}$. Thus, $\mathcal{B}$ can break the mode indistinguishability property of PPS with non-negligible probability. Thus, lemma holds. □

Claim 6.6. Let E be the event from Lemma 6.4. If PPS satisfies payload hiding, then $|\Pr[1 \leftarrow \mathbf{Hyb}^{\mathcal{A}}_{j-1}(1^\lambda) \wedge \overline{\mathsf{E}}] - \Pr[1 \leftarrow \mathbf{Hyb}^{\mathcal{A}}_{j}(1^\lambda) \wedge \overline{\mathsf{E}}]| \le \mathsf{negl}(\lambda)$.

Proof. Assume for the sake of contradiction that there exists an adversary $\mathcal{A}$ such that $|\Pr[1 \leftarrow \mathbf{Hyb}^{\mathcal{A}}_{j-1}(1^\lambda) \wedge \overline{\mathsf{E}}] - \Pr[1 \leftarrow \mathbf{Hyb}^{\mathcal{A}}_{j}(1^\lambda) \wedge \overline{\mathsf{E}}]| = \epsilon(\lambda)$ for a non-negligible function $\epsilon(\cdot)$. We will use $\mathcal{A}$ to construct a reduction adversary $\mathcal{B}$ that will break the payload hiding property of PPS with non-negligible advantage. Description of $\mathcal{B}$ is as follows.

$\mathcal{B}$ receives $1^{\ell_{\mathsf{out}}}$ from $\mathcal{A}$. It will send $(\ell_{\mathsf{out}}, j, \mu_j)$ to PPS payload hiding challenger where $\mu_j \leftarrow \{0,1\}$ sampled randomly by $\mathcal{B}$. $\mathcal{B}$ will receive $(\mathsf{pps.crs}, \mathsf{pps.ct}_j)$. $\mathcal{B}$ will then sample $\mu_i \leftarrow \{0,1\}$ for each $i > j$ and send (i, μ_i) to PPS challenger to receive $\mathsf{pps.ct}_i$. $\mathcal{B}$ will then sample $\mathsf{pps.ct}_i \leftarrow \mathsf{PPS.pkEnc}(i)$ for each $i < j$. It will set $\mathsf{pp} := (\mathsf{pps.crs}, (\mathsf{pps.ct}_i)_{i \in [\ell_{\mathsf{out}}]})$ and send pp to $\mathcal{A}$. $\mathcal{A}$ will then send $S \subseteq [\ell_{\mathsf{out}}]$ such that $j \notin S$. $\mathcal{B}$ will pass this S to PPS challenger and receive $\mathsf{pps.sk}_S$. $\mathcal{B}$ will set $\mathsf{sk}_S := \mathsf{pps.sk}_S$ and send sk_S to $\mathcal{A}$. $\mathcal{B}$ will output $\mathcal{A}$'s guess b' as its own.

Firstly, note that $\mathcal{B}$ is a valid adversary against the payload hiding game. If PPS challenger samples $\mathsf{pps.ct}_j$ using $\mathsf{PPS.skEnc}$, then $\mathcal{B}$ behaves like $\mathbf{Hyb}^{\mathcal{A}}_{j-1}(1^\lambda) \wedge \overline{\mathsf{E}}$ and if PPS challenger samples $\mathsf{pps.ct}_j$ using $\mathsf{PPS.pkEnc}$, then $\mathcal{B}$ behaves like $\mathbf{Hyb}^{\mathcal{A}}_{j}(1^\lambda) \wedge \overline{\mathsf{E}}$. Thus, $\mathcal{B}$ can break the payload hiding property of PPS with non-negligible probability. Thus, the lemma holds. □

Using these claims, we prove the lemma. □

Lemma 6.7. $\Pr[1 \leftarrow \mathbf{Hyb}^{\mathcal{A}}_{\ell_{\mathsf{out}}}(1^\lambda)] = \Pr[1 \leftarrow \mathbf{Hyb}^{\mathcal{A}}_{\ell_{\mathsf{out}}+1}(1^\lambda)]$.

Proof. The only difference between these two lemmas is that we will use an additional $\mathsf{pps.msk}' \leftarrow \mathsf{PPS.MSKGen}$ to generate $\mathsf{pps.sk}'_S \leftarrow \mathsf{PPS.KGen}(\mathsf{pps.msk}', S)$ during challenge phase. Note that the original $\mathsf{pps.msk}$ is not used anywhere in the experiment as all $\mathsf{pps.ct}_i$ are sampled via $\mathsf{PPS.pkEnc}(i)$. Thus, sampling an additional $\mathsf{pps.msk}'$ will not change the distribution of sk_S in any way. Hence, the lemma holds. □

Lemma 6.8. If PPS satisfies mode indistinguishability (Definition 4.1), then for any $j \in [\ell_{\mathsf{out}}]$, any PPT adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for any $\lambda \in \mathbb{N}$, $|\Pr[1 \leftarrow \mathbf{Hyb}^{\mathcal{A}}_{\ell_{\mathsf{out}}+1+j-1}(1^\lambda)] - \Pr[1 \leftarrow \mathbf{Hyb}^{\mathcal{A}}_{\ell_{\mathsf{out}}+1+j}(1^\lambda)]| \leq \mathsf{negl}(\lambda)$.

Proof. Note that we do not require payload hiding here. Assume for the sake of contradiction that there exists an adversary $\mathcal{A}$ such that $|\Pr[1 \leftarrow \mathbf{Hyb}^{\mathcal{A}}_{\ell_{\mathsf{out}}+1+j-1}(1^\lambda)] - \Pr[1 \leftarrow \mathbf{Hyb}^{\mathcal{A}}_{\ell_{\mathsf{out}}+1+j}(1^\lambda)]| = \epsilon(\lambda)$ for a non-negligible function $\epsilon(\cdot)$. We will use $\mathcal{A}$ to construct a reduction adversary $\mathcal{B}$ that will break the mode indistinguishability property of PPS with non-negligible advantage. Description of $\mathcal{B}$ is as follows.

$\mathcal{B}$ receives $1^{\ell_{\mathsf{out}}}$ from $\mathcal{A}$. It will send (ℓ_{out}, j) to PPS mode indistinguishability challenger. $\mathcal{B}$ will receive $(\mathsf{pps.crs}, \mathsf{pps.ct}_j)$. $\mathcal{B}$ will then sample $\mu_i \leftarrow \{0,1\}$ for each $i < j$ and send (i, μ_i) to PPS challenger to receive $\mathsf{pps.ct}_i$. $\mathcal{B}$ will then sample $\mathsf{pps.ct}_i \leftarrow \mathsf{PPS.pkEnc}(i)$ for each $i > j$. It will set $\mathsf{pp} := (\mathsf{pps.crs}, (\mathsf{pps.ct}_i)_{i \in [\ell_{\mathsf{out}}]})$ and send pp to $\mathcal{A}$. $\mathcal{A}$ will then send $S \subseteq [\ell_{\mathsf{out}}]$. $\mathcal{B}$ will sample $\mathsf{pps.msk}' \leftarrow \mathsf{PPS.MSKGen}$ and $\mathsf{pps.sk}'_S \leftarrow \mathsf{PPS.KGen}(\mathsf{pps.msk}', S)$. $\mathcal{B}$ will send $\{j\}$ to PPS challenger and receive $\mathsf{pps.sk}_{\{j\}}$. $\mathcal{B}$ will discard $\mathsf{pps.sk}_{\{j\}}$. $\mathcal{B}$ will set $\mathsf{sk}_S := \mathsf{pps.sk}'_S$ and send sk_S to $\mathcal{A}$. $\mathcal{B}$ will output $\mathcal{A}$'s guess b' as its own.

Firstly, note that $\mathcal{B}$ is a valid adversary against the mode indistinguishability game. If PPS challenger samples $\mathsf{pps.ct}_j$ using $\mathsf{PPS.pkEnc}$, then $\mathcal{B}$ behaves like $\mathbf{Hyb}^{\mathcal{A}}_{\ell_{\mathsf{out}}+1+j-1}(1^\lambda)$ and if PPS challenger samples $\mathsf{pps.ct}_j$ using $\mathsf{PPS.skEnc}$, then $\mathcal{B}$ behaves like $\mathbf{Hyb}^{\mathcal{A}}_{\ell_{\mathsf{out}}+1+j}(1^\lambda)$. Thus, $\mathcal{B}$ can break the mode indistinguishability property of PPS with non-negligible probability. Thus, lemma holds. □

Using these lemmas, we prove the theorem. □

Adaptive Pseudorandomness Analysis

Theorem 6.9 (Adaptive Pseudorandomness)**.** If PPS satisfies payload hiding property (Definition 4.1), then Construction 6.2 satisfies adaptive pseudorandomness property (Definition 6.1).

Proof. We prove this using the following hybrids for an admissible adversary $\mathcal{A}$. Note that we do not need mode indistinguishability property of PPS here. Note that $\mathcal{A}$ makes at most one projection query.

$\mathbf{Hyb}_j^{\mathcal{A}}(1^\lambda)$ for $j \in [0, \ell_{\mathsf{out}}]$. This is the adaptive pseudorandomness experiment from Definition 6.1 with $b = 0$ except that given an evaluation query on index $i \in [\ell_{\mathsf{out}}]$, if $i \leq j$, we send y_i to $\mathcal{A}$ where $\mathbf{y} \leftarrow \{0,1\}^{\ell_{\mathsf{out}}}$ is sampled during the setup phase, and otherwise we send μ_i sampled during the setup phase. By correctness of PPS, $\mathsf{Eval}([\ell_{\mathsf{out}}], \mathsf{sk}_{[\ell_{\mathsf{out}}]}, i) = \mu_i$ for each $i \in [\ell_{\mathsf{out}}]$. So, $\mathbf{Hyb}_0^{\mathcal{A}}(1^\lambda)$ is the experiment with $b = 0$ and $\mathbf{Hyb}_{\ell_{\mathsf{out}}}^{\mathcal{A}}(1^\lambda)$ is the experiment with $b = 1$.

Lemma 6.10. If PPS satisfies payload hiding property (Definition 4.1), then for any $j \in [\ell_{\mathsf{out}}]$ and admissible stateful PPT adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for any $\lambda \in \mathbb{N}$, $|\Pr[1 \leftarrow \mathbf{Hyb}_{j-1}^{\mathcal{A}}(1^\lambda)] - \Pr[1 \leftarrow \mathbf{Hyb}_j^{\mathcal{A}}(1^\lambda)]| \leq \mathsf{negl}(\lambda)$.

Proof. Note that $\mathbf{Hyb}_{j-1}^{\mathcal{A}}(1^\lambda)$ and $\mathbf{Hyb}_j^{\mathcal{A}}(1^\lambda)$ only differ in the response to the evaluation query on j. So, both hybrids are identical on executions where $j \notin \mathcal{I}$ at the end of the experiment, and it suffices to bound the difference of the probabilities that the two hybrids output 1 and $j \in \mathcal{I}$ at the end. Moreover, as $\mathcal{A}$ is admissible, if $\mathcal{A}$ makes its projection query on a set S with $j \in S$, then $j \notin \mathcal{I}$ at the end. We use the following sub-hybrids.

$\mathbf{SubHyb}_0^{\mathcal{A}}(1^\lambda)$. This is the same as $\mathbf{Hyb}_{j-1}^{\mathcal{A}}(1^\lambda)$ except that the challenger outputs 0 at the end if $j \notin \mathcal{I}$ and stops and outputs 0 when $\mathcal{A}$ makes a projection query on a set containing j. Note that $\mathbf{SubHyb}_0^{\mathcal{A}}(1^\lambda)$ outputs 1 if and only if $\mathbf{Hyb}_{j-1}^{\mathcal{A}}(1^\lambda)$ outputs 1 and $j \in \mathcal{I}$ at the end.

$\mathbf{SubHyb}_1^{\mathcal{A}}(1^\lambda)$. This is the same as $\mathbf{SubHyb}_0^{\mathcal{A}}(1^\lambda)$ except that we sample $\mathsf{pps.ct}_j \leftarrow \mathsf{PPS.pkEnc}(j)$ during the setup phase. We still answer the evaluation query on j with μ_j.

$\mathbf{SubHyb}_2^{\mathcal{A}}(1^\lambda)$. This is the same as $\mathbf{SubHyb}_1^{\mathcal{A}}(1^\lambda)$ except that we answer the evaluation query on j with y_j.

$\mathbf{SubHyb}_3^{\mathcal{A}}(1^\lambda)$. This is the same as $\mathbf{SubHyb}_2^{\mathcal{A}}(1^\lambda)$ except that we sample $\mathsf{pps.ct}_j \leftarrow \mathsf{PPS.skEnc}(\mathsf{pps.msk}, j, \mu_j)$ during the setup phase. Note that $\mathbf{SubHyb}_3^{\mathcal{A}}(1^\lambda)$ outputs 1 if and only if $\mathbf{Hyb}_j^{\mathcal{A}}(1^\lambda)$ outputs 1 and $j \in \mathcal{I}$ at the end.

Claim 6.11. If PPS satisfies payload hiding property, then for any admissible stateful PPT adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for any $\lambda \in \mathbb{N}$, $|\Pr[1 \leftarrow \mathbf{SubHyb}_0^{\mathcal{A}}(1^\lambda)] - \Pr[1 \leftarrow \mathbf{SubHyb}_1^{\mathcal{A}}(1^\lambda)]| \leq \mathsf{negl}(\lambda)$ and $|\Pr[1 \leftarrow \mathbf{SubHyb}_2^{\mathcal{A}}(1^\lambda)] - \Pr[1 \leftarrow \mathbf{SubHyb}_3^{\mathcal{A}}(1^\lambda)]| \leq \mathsf{negl}(\lambda)$.

Proof. Assume for the sake of contradiction that there exists an adversary $\mathcal{A}$ such that $|\Pr[1 \leftarrow \mathbf{SubHyb}_0^{\mathcal{A}}(1^\lambda)] - \Pr[1 \leftarrow \mathbf{SubHyb}_1^{\mathcal{A}}(1^\lambda)]| = \epsilon(\lambda)$ for a non-negligible function $\epsilon(\cdot)$. We will construct a reduction adversary $\mathcal{B}$ that will break the payload hiding property of PPS with non-negligible probability.

$\mathcal{B}$ receives $1^{\ell_{\mathsf{out}}}$ from $\mathcal{A}$. $\mathcal{B}$ samples $\mu_j \leftarrow \{0,1\}$ and sends $(\ell_{\mathsf{out}}, j, \mu_j)$ to the PPS payload hiding challenger. $\mathcal{B}$ receives $\mathsf{pps.crs}, \mathsf{pps.ct}_j$. $\mathcal{B}$ samples $\mu_i \leftarrow \{0,1\}$ for each $i \in [\ell_{\mathsf{out}}] \setminus \{j\}$ and queries the PPS challenger with (i, μ_i) to receive $\mathsf{pps.ct}_i$. $\mathcal{B}$ also samples $\mathbf{y} \leftarrow \{0,1\}^{\ell_{\mathsf{out}}}$. $\mathcal{B}$ sets $\mathsf{pp} := (\mathsf{pps.crs}, (\mathsf{pps.ct}_i)_{i\in[\ell_{\mathsf{out}}]})$, sends pp to $\mathcal{A}$, and answers the queries of $\mathcal{A}$ as follows.

- Given an evaluation query on $i \in [\ell_{\mathsf{out}}]$, $\mathcal{B}$ sends y_i to $\mathcal{A}$ if $i < j$ and μ_i otherwise.
- Given the projection query $S \subseteq [\ell_{\mathsf{out}}]$, if $j \in S$, $\mathcal{B}$ stops and outputs 0. Otherwise, $\mathcal{B}$ sends S to the PPS challenger, receives $\mathsf{pps.sk}_S$, and sends $\mathsf{sk}_S := \mathsf{pps.sk}_S$ to $\mathcal{A}$.

At the end, if $\mathcal{B}$ has not sent any set to the PPS challenger, it sends $\emptyset$ and ignores the response. If $j \notin \mathcal{I}$, $\mathcal{B}$ outputs 0. Otherwise, $\mathcal{B}$ outputs the guess b' of $\mathcal{A}$ as its own.

$\mathcal{B}$ sends at most one set to the PPS challenger, this set does not contain j, and $\mathcal{B}$ makes all its oracle queries before sending it. So, $\mathcal{B}$ is a valid payload hiding adversary against PPS. If the PPS challenger samples $\mathsf{pps.ct}_j$ using $\mathsf{PPS.skEnc}$, then $\mathcal{B}$ behaves like $\mathbf{SubHyb}_0^{\mathcal{A}}(1^\lambda)$. Otherwise, it behaves like $\mathbf{SubHyb}_1^{\mathcal{A}}(1^\lambda)$. Thus, $\mathcal{B}$ breaks the payload hiding property of PPS with advantage $\epsilon(\lambda)$. The argument for $\mathbf{SubHyb}_2^{\mathcal{A}}(1^\lambda)$ and $\mathbf{SubHyb}_3^{\mathcal{A}}(1^\lambda)$ is the same except that $\mathcal{B}$ answers the evaluation query on j with y_j. Thus, the claim holds. □

Claim 6.12. $\Pr[1 \leftarrow \mathbf{SubHyb}_1^{\mathcal{A}}(1^\lambda)] = \Pr[1 \leftarrow \mathbf{SubHyb}_2^{\mathcal{A}}(1^\lambda)]$.

Proof. In $\mathbf{SubHyb}_1^{\mathcal{A}}(1^\lambda)$, $\mathsf{pps.ct}_j$ is sampled using $\mathsf{PPS.pkEnc}$. So, μ_j is only used to answer the evaluation query on j, and this response is a uniformly random bit independent of all other values, exactly like y_j in $\mathbf{SubHyb}_2^{\mathcal{A}}(1^\lambda)$. Thus, the claim holds. □

Using these claims, we prove the lemma. □

Using this lemma, we prove the theorem. □

Remark 6.13 (PPS with weaker security and PSPRG)**.** Although we do not write it explicitly, it is possible to use the weaker security definition for PPS of [GY26c, Definition 6.1] to show the implication from PPS to PSPRG scheme (by substituting it in Construction 6.2). However, constructions of PPS from [GY26c] are inefficient or use exotic lattice assumptions which will yield inefficient constructions of PSPRG from LWE that are worse than the LWE-based construction of [HWW25]. Thus, our work shows that PPS with optimal efficiency is a stronger object that is realizable from standard LWE assumption and can be used to remove random oracles in various contexts.

6.2 Succinct CSS for Graph Access Structures from Robust PPS

In this subsection, we construct succinct CSS schemes for (undirected) graph access structures. The authorized sets in the access structure of a graph are the sets of vertices that contain at least one edge, and the unauthorized sets are the independent sets of the graph, namely the sets that contain no edge. Our construction follows one template. Every vertex v holds a secret key for the set of its neighbors and a ciphertext of the secret under the index v. For an edge $\{u, v\}$, the secret key of v decrypts the ciphertext of u, whereas the secret keys of an independent set decrypt none of its ciphertexts. Figure 2 illustrates this. The primitive we need is a generalization and relaxation of PPS whose payload hiding property holds against any number of secret keys. We call it a robust

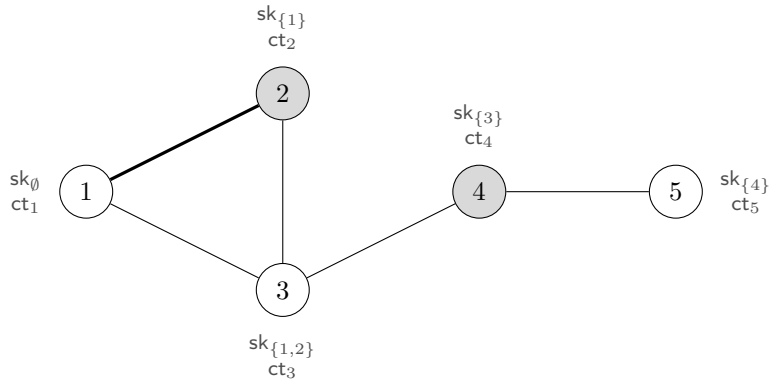

Figure 2: CSS for the access structure of a graph on five vertices that joined in the order $1, \ldots, 5$. Every vertex v holds a secret key sk_{S_v} for the set S_v of its neighbors that joined before it, and a ciphertext ct_v of the secret under the index v. Along the edge $\{1, 2\}$, $\mathsf{sk}_{\{1\}}$ decrypts ct_1. The shaded vertices form an independent set and the keys $\mathsf{sk}_{\{1\}}$ and $\mathsf{sk}_{\{3\}}$ decrypt neither ct_2 nor ct_4.

PPS and define it first. We then define CSS for graphs where the vertices join one at a time, and construct this from robust PPS.

Robust PPS. A robust programmable pseudorandom set sampler (robust PPS) scheme for message space $\mathcal{M} = \{\mathcal{M}_\lambda\}_{\lambda \in \mathbb{N}}$ consists of the following polynomial time algorithms.

$\mathsf{Setup}(1^\lambda, N) \to (\mathsf{pp}, \mathsf{msk})$**.** The randomized setup algorithm takes as input security parameter λ, number of indices N (in binary), and outputs public parameters pp and master secret key msk. The following algorithms take pp implicitly.

$\mathsf{pkEnc}(i) \to \mathsf{ct}$**.** The randomized *public-key* encryption algorithm takes as input index $i \in [N]$ and outputs ciphertext ct.

$\mathsf{KGen}(\mathsf{msk}, S) \to \mathsf{sk}_S$**.** The possibly randomized secret key generation algorithm takes as input master secret key msk, set $S \subseteq [N]$, and outputs secret key sk_S.

$\mathsf{skEnc}(\mathsf{msk}, i, \mu) \to \mathsf{ct}$**.** The randomized *secret-key* encryption algorithm takes as input master secret key msk, index $i \in [N]$, message $\mu \in \mathcal{M}_\lambda$, and outputs ciphertext ct.

$\mathsf{Decode}(S, \mathsf{sk}_S, i, \mathsf{ct}) \to \mu'$**.** The deterministic decoding algorithm takes as input set $S \subseteq [N]$, secret key sk_S, index $i \in S$, ciphertext ct, and outputs $\mu' \in \mathcal{M}_\lambda \cup \{\perp\}$.

Definition 6.14 (Robust PPS)**.** An RPPS scheme $(\mathsf{Setup}, \mathsf{pkEnc}, \mathsf{KGen}, \mathsf{skEnc}, \mathsf{Decode})$ is said to be a robust programmable pseudorandom set sampler scheme (robust PPS) if it satisfies the following properties.

Correctness. There exists a negligible function $\mathsf{negl}(\cdot)$ such that for any $\lambda \in \mathbb{N}$, $N \leq 2^\lambda$, $S \subseteq [N]$, $i \in S$, and $\mu \in \mathcal{M}_\lambda$,

$$\Pr\left[\mu = \mathsf{Decode}(S, \mathsf{sk}_S, i, \mathsf{ct}) \quad : \quad \begin{array}{l}(\mathsf{pp}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda, N),\\ \mathsf{sk}_S \leftarrow \mathsf{KGen}(\mathsf{msk}, S),\\ \mathsf{ct} \leftarrow \mathsf{skEnc}(\mathsf{msk}, i, \mu)\end{array}\right] \geq 1 - \mathsf{negl}(\lambda)$$

Compactness. There exist polynomials $\mathsf{poly}_{\mathsf{sk}}(\cdot,\cdot)$ and $\mathsf{poly}_{\mathsf{ct}}(\cdot,\cdot)$ such that for any $\lambda \in \mathbb{N}$, $N \leq 2^\lambda$, $S \subseteq [N]$, $i \in [N]$, and $\mu \in \mathcal{M}_\lambda$, $|\mathsf{sk}_S| = \mathsf{poly}_{\mathsf{sk}}(\lambda, \log N)$ and $|\mathsf{ct}| = \mathsf{poly}_{\mathsf{ct}}(\lambda, \log N)$.

Robust Payload Hiding. For any stateful admissible PPT adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for any $\lambda \in \mathbb{N}$,

$$\left|\Pr\left[\begin{array}{rcl} & & (N, i^* \in [N], \mu_{i^*} \in \mathcal{M}_\lambda) \leftarrow \mathcal{A}(1^\lambda), b \leftarrow \{0,1\}, \\ b \leftarrow \mathcal{A}^{\mathcal{O}(\cdot,\cdot)}(\mathsf{pp}, \mathsf{ct}_{i^*,b}) & : & (\mathsf{pp}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda, N), \\ & & \text{if } b = 0, \mathsf{ct}_{i^*,0} \leftarrow \mathsf{pkEnc}(i^*), \\ & & \text{if } b = 1, \mathsf{ct}_{i^*,1} \leftarrow \mathsf{skEnc}(\mathsf{msk}, i^*, \mu_{i^*}) \end{array}\right] - \frac{1}{2}\right| \leq \mathsf{negl}(\lambda)$$

where $\mathcal{O}$ is a stateful oracle that responds to the following queries.

(Enc, i, μ)**.** Given $i \in [N]$ and $\mu \in \mathcal{M}_\lambda$, $\mathcal{O}$ replies with $\mathsf{ct} \leftarrow \mathsf{skEnc}(\mathsf{msk}, i, \mu)$.

(KGen, S)**.** Given $S \subseteq [N]$, $\mathcal{O}$ replies with $\mathsf{sk}_S \leftarrow \mathsf{KGen}(\mathsf{msk}, S)$.

A stateful PPT machine $\mathcal{A}$ is said to be admissible if every query (Enc, i, μ) satisfies $i \neq i^*$ and every query (KGen, S) satisfies $i^* \notin S$.

Remark 6.15 (Post-key ciphertext queries)**.** In the payload hiding experiment of Definition 4.1, the oracle $\mathcal{O}$ is available only until $\mathcal{A}$ sends the set S. Construction 4.6 satisfies the stronger variant where $\mathcal{O}$ remains available after $\mathcal{A}$ receives sk_S. In every hybrid of the proof of Theorem 4.26, a query (i, μ_i) is answered with a fresh garbling of a fresh circuit C_i masked by the SEPRF at a fresh point t_i, and sk_S contains the actual PRF key, so the response does not depend on when the query is made. The abort on $t_i = t^*$ applies to all queries. The SEPRF reductions hold the challenge key throughout, the garbling reduction answers queries with its $\mathsf{CircGarb}(\mathsf{gk}, \cdot)$ oracle which Definition 4.4 provides after the challenge, and the LWE reduction samples the randomness of every C_i itself. The same holds for mode indistinguishability. In the rest of the paper, whenever we rely on payload hiding of PPS, we rely on this variant and refer to this remark.

6.2.1 CSS for Evolving Graphs

We consider the rigid evolving graphs of Francati and Venturi [FV25], where vertices arrive one at a time and existing shares are not modified. Our security game explicitly allows adaptive neighborhood choices and corruptions. The t-th vertex to join is named t and declares the set $S_t \subseteq [t-1]$ of existing vertices it is adjacent to. For sets $S_1, \ldots, S_L$ with $S_t \subseteq [t-1]$, let $G_L := ([L], E_L)$ where $E_L := \{\{u, t\} : t \in [L], u \in S_t\}$, and let $N(v)$ denote the set of neighbors of v in G_L, so that $S_t = N(t) \cap [t-1]$. A CSS scheme for evolving graphs with collusion bound Q and degree bound Δ consists of the following polynomial time algorithms.

$\mathsf{Setup}(1^\lambda, 1^Q, 1^\Delta, s) \to (\mathsf{sh}_0, \mathsf{st}_0)$**.** The randomized setup algorithm takes as input security parameter λ, collusion bound Q, degree bound Δ, secret $s \in \{0,1\}$, and outputs a public share sh_0 and an initial state st_0.

$\mathsf{AddVertex}(\mathsf{st}_{t-1}, S_t) \to (\mathsf{sh}_t, \mathsf{st}_t)$**.** The randomized vertex addition algorithm takes as input the state st_{t-1} after $t-1$ vertices have joined and a set $S_t \subseteq [t-1]$ of size at most Δ, and outputs a share sh_t for the new vertex t and an updated state st_t.

$\mathsf{Recon}(G, A, \mathsf{sh}_0, (\mathsf{sh}_v)_{v\in A}) \to s'$**.** The deterministic reconstruction algorithm takes as input graph G with vertex set $[L]$, a set $A \subseteq [L]$ that contains an edge of G, the public share sh_0, the shares of the vertices in A, and outputs $s' \in \{0,1\}$.

Definition 6.16 (CSS for Evolving Graphs)**.** A scheme $(\mathsf{Setup}, \mathsf{AddVertex}, \mathsf{Recon})$ is said to be an adaptively secure CSS scheme for evolving graphs with collusion bound Q and degree bound Δ if it satisfies the following properties.

Correctness. There exists a negligible function $\mathsf{negl}(\cdot)$ such that for any $\lambda, Q, \Delta, L \in \mathbb{N}$, $s \in \{0,1\}$, sets $S_1, \ldots, S_L$ with $S_t \subseteq [t-1]$ and $|S_t| \le \Delta$ for each $t \in [L]$, and set $A \subseteq [L]$ that contains an edge of G_L,

$$\Pr\left[\begin{array}{l} s = \mathsf{Recon}(G_L, A, \mathsf{sh}_0, (\mathsf{sh}_v)_{v\in A}) \end{array} : \begin{array}{l} (\mathsf{sh}_0, \mathsf{st}_0) \leftarrow \mathsf{Setup}(1^\lambda, 1^Q, 1^\Delta, s), \\ (\mathsf{sh}_t, \mathsf{st}_t) \leftarrow \mathsf{AddVertex}(\mathsf{st}_{t-1}, S_t) \\ \text{for } t = 1, \ldots, L \end{array}\right] \ge 1 - \mathsf{negl}(\lambda)$$

Adaptive Security. For any stateful admissible PPT adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for any $\lambda \in \mathbb{N}$,

$$\left|\Pr\left[\begin{array}{l} b \leftarrow \mathcal{A}^{\mathcal{O}(\cdot,\cdot)}(\mathsf{sh}_0) \end{array} : \begin{array}{l} \mathsf{ctr} := 0, \mathcal{C} := \emptyset, \\ (1^Q, 1^\Delta) \leftarrow \mathcal{A}(1^\lambda), b \leftarrow \{0,1\}, \\ (\mathsf{sh}_0, \mathsf{st}_0) \leftarrow \mathsf{Setup}(1^\lambda, 1^Q, 1^\Delta, b) \end{array}\right] - \frac{1}{2}\right| \le \mathsf{negl}(\lambda)$$

where $\mathcal{O}$ is a stateful oracle that updates $\mathsf{ctr}, \mathcal{C}$ and responds to the following queries.

- $(\mathsf{AddVertex}, S)$**.** Given $S \subseteq [\mathsf{ctr}]$ of size at most Δ, $\mathcal{O}$ updates $\mathsf{ctr} := \mathsf{ctr} + 1$, sets $S_{\mathsf{ctr}} := S$, samples $(\mathsf{sh}_{\mathsf{ctr}}, \mathsf{st}_{\mathsf{ctr}}) \leftarrow \mathsf{AddVertex}(\mathsf{st}_{\mathsf{ctr}-1}, S_{\mathsf{ctr}})$, and replies with ctr.
- $(\mathsf{Corrupt}, v)$**.** Given $v \in [\mathsf{ctr}]$, $\mathcal{O}$ replies with sh_v and updates $\mathcal{C} := \mathcal{C} \cup \{v\}$.

A stateful PPT machine $\mathcal{A}$ is said to be admissible if $|\mathcal{C}| \le Q$ and $\mathcal{C}$ is an independent set of G_L at the end of the experiment, where L is the final value of ctr. Note that $\mathcal{A}$ may declare edges from a new vertex to vertices in $\mathcal{C}$ and may corrupt a vertex after it has acquired neighbors.

Variants. We say that the scheme supports unbounded collusion if the running times of its algorithms and the share sizes grow with Q only through $\log Q$. In this case, we set $Q := 2^\lambda$, provide it to Setup in binary, and omit the collusion bound. Unbounded degree is defined in the same way with respect to Δ. For a graph G with vertex set $[L]$ that is fixed in advance, the dealer runs $\mathsf{Setup}(1^\lambda, 1^Q, 1^\Delta, s)$ for suitable bounds and then $\mathsf{AddVertex}(\mathsf{st}_{t-1}, N(t) \cap [t-1])$ for $t = 1, \ldots, L$, so that $G_L = G$. This gives a CSS scheme for the graph access structure of G in the sense of [ABI+23, Definitions 2.4 and 2.5] with the same share sizes, where the adversary additionally chooses the corrupted vertices adaptively.

Construction

We now construct a CSS scheme for evolving graphs using a robust PPS scheme RPPS (Definition 6.14) for a message space $\mathcal{M}$ with $\{0,1\} \subseteq \mathcal{M}_\lambda$.

Construction 6.17. We construct a CSS scheme for evolving graphs with unbounded collusion and unbounded degree as follows. The inputs 1^Q and 1^Δ are ignored.

$\mathsf{Setup}(1^\lambda, 1^Q, 1^\Delta, s)$**.** Sample $(\mathsf{pp}, \mathsf{msk}) \leftarrow \mathsf{RPPS.Setup}(1^\lambda, 2^\lambda)$. Output $\mathsf{sh}_0 := \mathsf{pp}$ and $\mathsf{st}_0 := (\mathsf{msk}, s)$.

$\mathsf{AddVertex}(\mathsf{st}_{t-1}, S_t)$**.** Parse st_{t-1} as (msk, s). Sample $\mathsf{sk}_t \leftarrow \mathsf{RPPS.KGen}(\mathsf{msk}, S_t)$ and $\mathsf{ct}_t \leftarrow \mathsf{RPPS.skEnc}(\mathsf{msk}, t, s)$. Output $\mathsf{sh}_t := (\mathsf{sk}_t, \mathsf{ct}_t)$ and $\mathsf{st}_t := \mathsf{st}_{t-1}$.

$\mathsf{Recon}(G, A, \mathsf{sh}_0, (\mathsf{sh}_v)_{v\in A})$**.** Parse sh_0 as pp and G as $([L], E)$. Let (u, v) be the lexicographically smallest pair in $A \times A$ such that $u < v$ and $\{u, v\} \in E$. Parse sh_u as $(\mathsf{sk}_u, \mathsf{ct}_u)$ and sh_v as $(\mathsf{sk}_v, \mathsf{ct}_v)$, and let $S_v := N(v) \cap [v-1]$.

Output $s' := \mathsf{RPPS.Decode}(S_v, \mathsf{sk}_v, u, \mathsf{ct}_u)$.

Correctness. The pair (u, v) chosen by Recon is determined by G_L and A and satisfies $u < v$ and $\{u, v\} \in E_L$. So, $u \in S_v$. Moreover, $(\mathsf{pp}, \mathsf{sk}_v, \mathsf{ct}_u)$ is distributed as in the correctness experiment of RPPS with $N = 2^\lambda$, $S = S_v$, $i = u$, and $\mu = s$. Thus, Recon outputs s with probability at least $1 - \mathsf{negl}(\lambda)$.

Share Size. By compactness of RPPS with $N = 2^\lambda$, $|\mathsf{sh}_t| = \mathsf{poly}_{\mathsf{sk}}(\lambda, \lambda) + \mathsf{poly}_{\mathsf{ct}}(\lambda, \lambda) = \mathsf{poly}(\lambda)$ for each vertex t, which does not depend on the number of vertices, on $|S_t|$, or on the number of corrupted vertices, and $|\mathsf{sh}_0| = |\mathsf{pp}|$. The state of the dealer does not grow. $\mathsf{AddVertex}$ runs in time $\mathsf{poly}(\lambda, |S_t|)$, where the dependence on $|S_t|$ comes only from $\mathsf{RPPS.KGen}$. So, the scheme supports unbounded collusion and unbounded degree.

Remark 6.18 (Removing the public share)**.** When $|\mathsf{pp}| = \mathsf{poly}(\lambda, \log N)$, the dealer can append pp to every share and output an empty public share. Recon reads pp from the share of either endpoint of the edge it uses, so correctness is unchanged, and the view of $\mathcal{A}$ in the security experiment is unchanged as it receives pp in either case. The shares remain of size $\mathsf{poly}(\lambda)$. This applies to the robust PPS from ABE (Section 6.3) and to the batched IBE instantiation of [XC25], where pp has size $\mathsf{poly}(\lambda, \log N)$ and $\mathsf{poly}(\lambda, \log B)$ respectively. For the pairing based instantiations of batched IBE, pp consists of $O(B)$ group elements, and keeping it as a public share is what keeps the shares at $O(1)$ group elements.

Theorem 6.19. If RPPS satisfies robust payload hiding (Definition 6.14), then Construction 6.17 is an adaptively secure CSS scheme for evolving graphs (Definition 6.16) with unbounded collusion and unbounded degree where the shares have size $\mathsf{poly}(\lambda)$ and the public share consists of the public parameters of RPPS for 2^λ indices.

Proof. Let $Q_{\mathsf{AddVertex}} = Q_{\mathsf{AddVertex}}(\lambda)$ be a polynomial upper bound on the number of queries of the form $(\mathsf{AddVertex}, \cdot)$ that $\mathcal{A}$ makes. We use the following hybrids.

$\mathbf{Hyb}_0^{\mathcal{A}}(1^\lambda)$**.** This is the experiment of Definition 6.16 with $b = 0$.

Setup Phase. $\mathcal{A}$ sends $(1^Q, 1^\Delta)$. Sample $(\mathsf{pp}, \mathsf{msk}) \leftarrow \mathsf{RPPS.Setup}(1^\lambda, 2^\lambda)$ and send $\mathsf{sh}_0 := \mathsf{pp}$ to $\mathcal{A}$. Set $\mathsf{ctr} := 0$.

Query Phase. $\mathcal{A}$ adaptively makes the following queries.

- On a query $(\mathsf{AddVertex}, S)$ with $S \subseteq [\mathsf{ctr}]$, update $\mathsf{ctr} := \mathsf{ctr}+1$ and set $S_{\mathsf{ctr}} := S$. Sample $\mathsf{sk}_{\mathsf{ctr}} \leftarrow \mathsf{RPPS.KGen}(\mathsf{msk}, S_{\mathsf{ctr}})$ and $\mathsf{ct}_{\mathsf{ctr}} \leftarrow \mathsf{RPPS.skEnc}(\mathsf{msk}, \mathsf{ctr}, 0)$, set $\mathsf{sh}_{\mathsf{ctr}} := (\mathsf{sk}_{\mathsf{ctr}}, \mathsf{ct}_{\mathsf{ctr}})$, and return ctr.
- On a query $(\mathsf{Corrupt}, v)$ with $v \in [\mathsf{ctr}]$, return sh_v.

Guess Phase. $\mathcal{A}$ outputs its guess b'. Output b'.

$\mathbf{Hyb}_1^{\mathcal{A}}(1^\lambda)$. This is the same as $\mathbf{Hyb}_0^{\mathcal{A}}(1^\lambda)$ except that on a query $(\mathsf{AddVertex}, S)$, we only update $\mathsf{ctr} := \mathsf{ctr}+1$, set $S_{\mathsf{ctr}} := S$, and return ctr. On a query $(\mathsf{Corrupt}, v)$, if sh_v has not been sampled yet, we sample $\mathsf{sk}_v \leftarrow \mathsf{RPPS.KGen}(\mathsf{msk}, S_v)$ and $\mathsf{ct}_v \leftarrow \mathsf{RPPS.skEnc}(\mathsf{msk}, v, 0)$ and set $\mathsf{sh}_v := (\mathsf{sk}_v, \mathsf{ct}_v)$. Then, we return sh_v.

As the response to a query $(\mathsf{AddVertex}, S)$ *does not depend on* $\mathsf{sh}_{\mathsf{ctr}}$ *and the shares are sampled independently of each other given* $(\mathsf{pp}, \mathsf{msk})$*, the view of* $\mathcal{A}$ *is identical in* $\mathbf{Hyb}_0^{\mathcal{A}}(1^\lambda)$ *and* $\mathbf{Hyb}_1^{\mathcal{A}}(1^\lambda)$.

$\mathbf{Hyb}_{2,j}^{\mathcal{A}}(1^\lambda)$ for $j \in [0, Q_{\mathsf{AddVertex}}]$. This is the same as $\mathbf{Hyb}_1^{\mathcal{A}}(1^\lambda)$ except that on a query $(\mathsf{Corrupt}, v)$ with $v \le j$, we sample $\mathsf{ct}_v \leftarrow \mathsf{RPPS.skEnc}(\mathsf{msk}, v, 1)$. Note that $\mathbf{Hyb}_{2,0}^{\mathcal{A}}(1^\lambda)$ is the same as $\mathbf{Hyb}_1^{\mathcal{A}}(1^\lambda)$.

$\mathbf{Hyb}_3^{\mathcal{A}}(1^\lambda)$. This is the same as $\mathbf{Hyb}_0^{\mathcal{A}}(1^\lambda)$ except that on a query $(\mathsf{AddVertex}, S)$, we sample $\mathsf{ct}_{\mathsf{ctr}} \leftarrow \mathsf{RPPS.skEnc}(\mathsf{msk}, \mathsf{ctr}, 1)$. This is the experiment of Definition 6.16 with $b = 1$.

As $\mathcal{A}$ *makes at most* $Q_{\mathsf{AddVertex}}$ *queries of the form* $(\mathsf{AddVertex}, \cdot)$*, every vertex* v *satisfies* $v \le Q_{\mathsf{AddVertex}}$*, and the view of* $\mathcal{A}$ *is identical in* $\mathbf{Hyb}_{2,Q_{\mathsf{AddVertex}}}^{\mathcal{A}}(1^\lambda)$ *and* $\mathbf{Hyb}_3^{\mathcal{A}}(1^\lambda)$ *by the same argument as for* $\mathbf{Hyb}_0^{\mathcal{A}}(1^\lambda)$ *and* $\mathbf{Hyb}_1^{\mathcal{A}}(1^\lambda)$.

Lemma 6.20. If RPPS satisfies robust payload hiding (Definition 6.14), then for any $j \in [Q_{\mathsf{AddVertex}}]$ and admissible PPT adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for any $\lambda \in \mathbb{N}$, $|\Pr[1 \leftarrow \mathbf{Hyb}_{2,j-1}^{\mathcal{A}}(1^\lambda)] - \Pr[1 \leftarrow \mathbf{Hyb}_{2,j}^{\mathcal{A}}(1^\lambda)]| \le \mathsf{negl}(\lambda)$.

Proof. $\mathbf{Hyb}_{2,j-1}^{\mathcal{A}}(1^\lambda)$ and $\mathbf{Hyb}_{2,j}^{\mathcal{A}}(1^\lambda)$ differ only in the way ct_j is sampled, and ct_j is sampled only if $\mathcal{A}$ corrupts j. So, the two hybrids proceed identically as long as $\mathcal{A}$ does not corrupt j, and the probability that $\mathcal{A}$ outputs 1 without corrupting j is the same in both. For $i \in \{0,1\}$, let p_i denote the probability that $\mathcal{A}$ outputs 1 and corrupts j in $\mathbf{Hyb}_{2,j-1+i}^{\mathcal{A}}(1^\lambda)$. Then, $|\Pr[1 \leftarrow \mathbf{Hyb}_{2,j-1}^{\mathcal{A}}(1^\lambda)] - \Pr[1 \leftarrow \mathbf{Hyb}_{2,j}^{\mathcal{A}}(1^\lambda)]| = |p_0 - p_1|$.

Let $\mathbf{SubHyb}_j^{\mathcal{A}}(1^\lambda)$ be the same as $\mathbf{Hyb}_{2,j-1}^{\mathcal{A}}(1^\lambda)$ except that on a query $(\mathsf{Corrupt}, j)$, we sample $\mathsf{ct}_j \leftarrow \mathsf{RPPS.pkEnc}(j)$, and let p' denote the probability that $\mathcal{A}$ outputs 1 and corrupts j in $\mathbf{SubHyb}_j^{\mathcal{A}}(1^\lambda)$. We show that $|p_0 - p'|$ and $|p' - p_1|$ are both negligible. Assume for the sake of contradiction that $|p_\mu - p'| = \epsilon(\lambda)$ for some $\mu \in \{0,1\}$ and a non-negligible function $\epsilon(\cdot)$. We construct a reduction adversary $\mathcal{B}$ that breaks robust payload hiding of RPPS with advantage $\epsilon(\lambda)/2$. $\mathcal{B}$ receives $(1^Q, 1^\Delta)$ from $\mathcal{A}$ and sends $(2^\lambda, j, \mu)$ to the RPPS challenger. It receives $(\mathsf{pp}, \mathsf{ct}^*)$, sends $\mathsf{sh}_0 := \mathsf{pp}$ to $\mathcal{A}$, and sets $\mathsf{ctr} := 0$. On a query $(\mathsf{AddVertex}, S)$, $\mathcal{B}$ updates $\mathsf{ctr} := \mathsf{ctr}+1$, sets $S_{\mathsf{ctr}} := S$, and returns ctr. On a query $(\mathsf{Corrupt}, v)$ for which sh_v has not been sampled yet, $\mathcal{B}$ does the following. If $j \in S_v$, $\mathcal{B}$ aborts and outputs 0. Otherwise, $\mathcal{B}$ queries (KGen, S_v) to get sk_v. If $v = j$, it sets $\mathsf{ct}_v := \mathsf{ct}^*$. If $v < j$, it queries $(\mathsf{Enc}, v, 1)$ to get ct_v, and if $v > j$, it queries $(\mathsf{Enc}, v, 0)$ to get ct_v. It sets $\mathsf{sh}_v := (\mathsf{sk}_v, \mathsf{ct}_v)$ and returns sh_v. When $\mathcal{A}$ outputs its guess b', $\mathcal{B}$ outputs b' if $\mathcal{A}$ has corrupted j and 0 otherwise.

We first check that $\mathcal{B}$ is admissible. The target index j lies in $[2^\lambda]$. Every query $(\mathsf{Enc}, v, \cdot)$ satisfies $v \neq j$. Every query (KGen, S_v) satisfies $S_v \subseteq [2^\lambda]$ and is made only if $j \notin S_v$. Note that $j \notin S_j$ as $S_j \subseteq [j-1]$.

Now, consider an execution in which $\mathcal{A}$ corrupts j. As $\mathcal{A}$ is admissible, the set $\mathcal{C}$ of corrupted vertices is an independent set of G_L. So, no $v \in \mathcal{C}$ satisfies $j \in S_v$, and $\mathcal{B}$ never aborts in such an execution. The secret keys and the ciphertexts of the vertices other than j are distributed as in both hybrids. If $\mathsf{ct}^* \leftarrow \mathsf{RPPS.skEnc}(\mathsf{msk}, j, \mu)$, then $\mathcal{B}$ simulates $\mathbf{Hyb}^{\mathcal{A}}_{2,j-1+\mu}(1^\lambda)$ for $\mathcal{A}$, and if $\mathsf{ct}^* \leftarrow \mathsf{RPPS.pkEnc}(j)$, then $\mathcal{B}$ simulates $\mathbf{SubHyb}^{\mathcal{A}}_j(1^\lambda)$. So, the probability that $\mathcal{B}$ outputs 1 is p_μ in the former case and p' in the latter case. In an execution in which $\mathcal{A}$ does not corrupt j, $\mathcal{B}$ outputs 0 whether or not it aborts, which is consistent with the above. Thus, $\mathcal{B}$ breaks robust payload hiding of RPPS with advantage $|p_\mu - p'|/2 = \epsilon(\lambda)/2$. Hence, $|p_0 - p_1| \leq |p_0 - p'| + |p' - p_1| \leq \mathsf{negl}(\lambda)$ and the lemma holds. □

Using this lemma, we prove the theorem. □

Bounded Variants. The known instantiations of robust PPS come with bounds, which carry over to Construction 6.17 as follows. If $\mathsf{RPPS.Setup}$ additionally takes a bound 1^B and KGen only supports sets of size at most B, with compactness $\mathsf{poly}(\lambda, \log N, \log B)$, we run $\mathsf{RPPS.Setup}(1^\lambda, 2^\lambda, 1^\Delta)$ in Setup and obtain a CSS scheme with degree bound Δ and shares of size $\mathsf{poly}(\lambda, \log \Delta)$. In this case, the proof of Theorem 6.19 applies verbatim. If RPPS is a Q-bounded robust PPS (Definition A.1), we run $\mathsf{RPPS.Setup}(1^\lambda, 2^\lambda, 1^Q)$ in Setup and obtain a CSS scheme with collusion bound Q. The proof of Theorem 6.19 goes through with two changes in the proof of Lemma 6.20. The two steps through $\mathsf{pkEnc}(j)$ are merged into a single step from $\mathsf{skEnc}(\mathsf{msk}, j, 0)$ to $\mathsf{skEnc}(\mathsf{msk}, j, 1)$ using bounded adaptive robustness with $(\mu_0, \mu_1) := (0, 1)$, where $\mathcal{B}$ obtains ct_t for $t \neq j$ through queries $(\mathsf{Enc}, t, 1)$ for $t < j$ and $(\mathsf{Enc}, t, 0)$ for $t > j$. Moreover, $\mathcal{B}$ makes one query of the form $(\mathsf{KGen}, \cdot)$ per corrupted vertex, so at most Q of them.

6.3 Constructing Robust PPS

In this subsection, we construct robust PPS (Definition 6.14) from batched IBE and from ABE with succinct keys. In both cases, a secret key for a set S decrypts exactly the ciphertexts under the indices in S, and security holds against any number of secret keys.

Construction from Batched IBE

We begin with defining batched identity based encryption (batched IBE) [AFP25].

Batched IBE. A batched IBE scheme for identity space $\mathcal{I} = \{\mathcal{I}_\lambda\}_{\lambda \in \mathbb{N}}$ consists of the following polynomial time algorithms. We assume that $[2^\lambda] \subseteq \mathcal{I}_\lambda$.

$\mathsf{Setup}(1^\lambda, 1^B) \to (\mathsf{mpk}, \mathsf{msk})$**.** The randomized setup algorithm takes as input security parameter λ, maximum batch size B, and outputs master public key mpk and master secret key msk.

$\mathsf{Digest}(\mathsf{mpk}, S) \to \mathsf{dig}$**.** The deterministic digest algorithm takes as input master public key mpk, a set $S \subseteq \mathcal{I}_\lambda$ of size at most B, and outputs digest dig.

$\mathsf{KGen}(\mathsf{msk}, \mathsf{dig}) \to \mathsf{sk}$**.** The randomized key generation algorithm takes as input master secret key msk, digest dig, and outputs secret key sk.

$\mathsf{Enc}(\mathsf{mpk}, \mathsf{id}, \mu) \to \mathsf{ct}$**.** The randomized encryption algorithm takes as input master public key mpk, identity $\mathsf{id} \in \mathcal{I}_\lambda$, message $\mu \in \{0,1\}$, and outputs ciphertext ct.

$\mathsf{Dec}(\mathsf{mpk}, S, \mathsf{sk}, \mathsf{id}, \mathsf{ct}) \to \mu'$**.** The deterministic decryption algorithm takes as input master public key mpk, a set $S \subseteq \mathcal{I}_\lambda$ of size at most B, secret key sk, identity $\mathsf{id} \in \mathcal{I}_\lambda$, ciphertext ct, and outputs $\mu' \in \{0,1\} \cup \{\bot\}$.

Definition 6.21 (Selectively Secure Batched IBE, [AFP25], adapted.)**.** A scheme $(\mathsf{Setup}, \mathsf{Digest}, \mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$ is said to be a selectively secure batched IBE scheme for identity space $\mathcal{I}$ if it satisfies the following properties.

Correctness. There exists a negligible function $\mathsf{negl}(\cdot)$ such that for any $\lambda, B \in \mathbb{N}$, $S \subseteq \mathcal{I}_\lambda$ of size at most B, $\mathsf{id} \in S$, and $\mu \in \{0,1\}$,

$$\Pr\left[\mu = \mathsf{Dec}(\mathsf{mpk}, S, \mathsf{sk}, \mathsf{id}, \mathsf{ct}) \quad : \quad \begin{array}{l}(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda, 1^B),\\ \mathsf{sk} \leftarrow \mathsf{KGen}(\mathsf{msk}, \mathsf{Digest}(\mathsf{mpk}, S)),\\ \mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{mpk}, \mathsf{id}, \mu)\end{array}\right] \geq 1 - \mathsf{negl}(\lambda)$$

Compactness. There exist polynomials $\mathsf{poly}_{\mathsf{sk}}$ and $\mathsf{poly}_{\mathsf{ct}}$ such that for any $\lambda, B \in \mathbb{N}$, we have $|\mathsf{sk}| = \mathsf{poly}_{\mathsf{sk}}(\lambda, \log B)$ and $|\mathsf{ct}| = \mathsf{poly}_{\mathsf{ct}}(\lambda, \log B)$.

Selective Security. For any stateful admissible PPT adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for any $\lambda \in \mathbb{N}$,

$$\left|\Pr\left[b \leftarrow \mathcal{A}^{\mathcal{O}}(\mathsf{ct}^*) \quad : \quad \begin{array}{l}(1^B, \mathsf{id}^*) \leftarrow \mathcal{A}(1^\lambda),\\ (\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda, 1^B),\\ (\mu_0, \mu_1) \leftarrow \mathcal{A}^{\mathcal{O}}(\mathsf{mpk}), b \leftarrow \{0,1\},\\ \mathsf{ct}^* \leftarrow \mathsf{Enc}(\mathsf{mpk}, \mathsf{id}^*, \mu_b)\end{array}\right] - \frac{1}{2}\right| \leq \mathsf{negl}(\lambda)$$

where the oracle $\mathcal{O}$ on input a set S outputs $\mathsf{KGen}(\mathsf{msk}, \mathsf{Digest}(\mathsf{mpk}, S))$. A stateful PPT machine $\mathcal{A}$ is said to be admissible if $\mathsf{id}^* \in \mathcal{I}_\lambda$ and every set S that it queries to $\mathcal{O}$ satisfies $S \subseteq \mathcal{I}_\lambda$, $|S| \leq B$, and $\mathsf{id}^* \notin S$. This condition applies both before and after receiving ct^*.

As batched IBE fixes the batch size B at setup, we obtain a robust PPS whose Setup additionally takes 1^B and whose KGen only supports sets of size at most B, as discussed in the bounded variants of Section 6.2.1.

Construction 6.22. We construct a robust PPS scheme for $\mathcal{M}_\lambda = \{0,1\}$ and sets of size at most B using a selectively secure batched IBE scheme BIBE (Definition 6.21) as follows.

$\mathsf{Setup}(1^\lambda, N, 1^B)$**.** Sample $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{BIBE.Setup}(1^\lambda, 1^B)$. Output $\mathsf{pp} := \mathsf{mpk}$ and msk.

$\mathsf{pkEnc}(i)$**.** Output $\mathsf{ct} \leftarrow \mathsf{BIBE.Enc}(\mathsf{mpk}, i, 0)$.

$\mathsf{KGen}(\mathsf{msk}, S)$**.** Given $S \subseteq [N]$ of size at most B, compute $\mathsf{dig} := \mathsf{BIBE.Digest}(\mathsf{mpk}, S)$. Output $\mathsf{sk}_S \leftarrow \mathsf{BIBE.KGen}(\mathsf{msk}, \mathsf{dig})$.

$\mathsf{skEnc}(\mathsf{msk}, i, \mu)$**.** Output $\mathsf{ct} \leftarrow \mathsf{BIBE.Enc}(\mathsf{mpk}, i, \mu)$.

$\mathsf{Decode}(S, \mathsf{sk}_S, i, \mathsf{ct})$**.** If $i \notin S$, output $\bot$. Otherwise, output $\mathsf{BIBE.Dec}(\mathsf{mpk}, S, \mathsf{sk}_S, i, \mathsf{ct})$.

Correctness. For $S \subseteq [N]$ of size at most B, $i \in S$, and $\mu \in \{0,1\}$, the tuple $(\mathsf{mpk}, \mathsf{sk}_S, \mathsf{ct})$ is distributed as in the correctness experiment of BIBE with S, $\mathsf{id} = i$, and μ, where $i \in [N] \subseteq [2^\lambda] \subseteq \mathcal{I}_\lambda$. So, correctness holds with probability at least $1 - \mathsf{negl}(\lambda)$.

Compactness. By compactness of BIBE, $|\mathsf{sk}_S| = \mathsf{poly}_{\mathsf{sk}}(\lambda, \log B)$ and $|\mathsf{ct}| = \mathsf{poly}_{\mathsf{ct}}(\lambda, \log B)$, which is $\mathsf{poly}(\lambda, \log N, \log B)$. Moreover, $|\mathsf{pp}| = |\mathsf{mpk}|$.

Theorem 6.23. If BIBE is a selectively secure batched IBE scheme (Definition 6.21), then Construction 6.22 satisfies robust payload hiding (Definition 6.14) for sets of size at most B.

Proof. Assume for the sake of contradiction that there exists an admissible adversary $\mathcal{A}$ with advantage $\epsilon(\lambda)$ in the robust payload hiding experiment for a non-negligible function $\epsilon(\cdot)$. We construct a reduction adversary $\mathcal{B}$ that breaks selective security of BIBE with the same advantage. $\mathcal{B}$ receives (N, i^*, μ_{i^*}) from $\mathcal{A}$ and sends $(1^B, i^*)$ to the BIBE challenger. It receives mpk, sends $(0, \mu_{i^*})$ to the challenger, and receives ct^*. It sends $(\mathsf{mpk}, \mathsf{ct}^*)$ to $\mathcal{A}$. On a query (Enc, i, μ), $\mathcal{B}$ replies with $\mathsf{BIBE.Enc}(\mathsf{mpk}, i, \mu)$. On a query (KGen, S), $\mathcal{B}$ queries S to $\mathcal{O}$ and replies with the response. $\mathcal{B}$ outputs the guess b' of $\mathcal{A}$ as its own.

As $\mathcal{A}$ is admissible, every set S that $\mathcal{B}$ queries to $\mathcal{O}$ satisfies $S \subseteq [N] \subseteq \mathcal{I}_\lambda$, $|S| \leq B$, and $i^* \notin S$. So, $\mathcal{B}$ is admissible. $\mathcal{B}$ answers the queries of the form $(\mathsf{Enc}, \cdot, \cdot)$ using mpk only. If ct^* encrypts 0, then ct^* is distributed as $\mathsf{pkEnc}(i^*)$ and $\mathcal{B}$ simulates the robust payload hiding experiment with $b = 0$ for $\mathcal{A}$. Otherwise, ct^* is distributed as $\mathsf{skEnc}(\mathsf{msk}, i^*, \mu_{i^*})$ and $\mathcal{B}$ simulates the experiment with $b = 1$. Thus, $\mathcal{B}$ breaks selective security of BIBE with advantage $\epsilon(\lambda)$ and the theorem holds. □

Instantiations. The batched IBE schemes of [AFP25, GWWW26] from pairings are selectively secure with secret keys and ciphertexts consisting of $O(1)$ group elements and master public keys consisting of $O(B)$ group elements, where [AFP25] is proven secure in the generic bilinear group and random oracle models and [GWWW26] under a q-type assumption in the plain model. Their identity space is $\mathbb{Z}_p$ for a prime p that is exponential in λ, so $[2^\lambda]$ embeds into it. The batched IBE scheme of [XC25] is selectively secure assuming ℓ-succinct LWE [Wee24] for $\ell = \mathsf{poly}(\lambda, \log B)$ with $|\mathsf{mpk}|, |\mathsf{sk}|, |\mathsf{ct}| = \mathsf{poly}(\lambda, \log B)$, once its batch labels are fixed to a constant, and its identity space is $\mathbb{Z}_q$ for a modulus q that is exponential in λ. All known schemes fix the batch size B at setup and their master public keys grow with B or $\log B$. Plugging them into Construction 6.22 and then into Construction 6.17 with $B := \Delta$, Theorem 6.23 and Theorem 6.19 give the following.

Corollary 6.24. Assuming the q-type assumption of [GWWW26] on bilinear groups, for any polynomial Δ, there exists an adaptively secure CSS scheme for evolving graphs with degree bound Δ and unbounded collusion where the shares consist of $O(1)$ group elements and the public share consists of $O(\Delta)$ group elements. The same holds in the generic bilinear group and random oracle models via [AFP25]. Assuming ℓ-succinct LWE [Wee24] for $\ell = \mathsf{poly}(\lambda, \log \Delta)$, there exists such a scheme where the shares have size $\mathsf{poly}(\lambda, \log \Delta)$ and there is no public share via [XC25] and Remark 6.18.

For a graph with L vertices that is fixed in advance, we set $\Delta := L$ and add the vertices in order. For a bipartite graph with parts L and R, we number the vertices in R before those in L and set Δ to the maximum degree of a vertex in L.

Construction from ABE with Succinct Keys

We begin with defining attribute-based encryption [SW05, GPSW06].

Attribute-Based Encryption. For $\ell, d \in \mathbb{N}$, let $\mathcal{F}_{\ell,d}$ denote the set of boolean circuits with ℓ-bit inputs and depth at most d. A key-policy ABE scheme for circuits consists of the following polynomial time algorithms.

$\mathsf{Setup}(1^\lambda, 1^\ell, 1^d) \to (\mathsf{mpk}, \mathsf{msk})$**.** The randomized setup algorithm takes as input security parameter λ, attribute length ℓ, depth bound d, and outputs master public key mpk and master secret key msk.

$\mathsf{KGen}(\mathsf{msk}, f) \to \mathsf{sk}_f$**.** The randomized key generation algorithm takes as input master secret key msk, circuit $f \in \mathcal{F}_{\ell,d}$, and outputs secret key sk_f.

$\mathsf{Enc}(\mathsf{mpk}, x, \mu) \to \mathsf{ct}$**.** The randomized encryption algorithm takes as input master public key mpk, attribute $x \in \{0,1\}^\ell$, message $\mu \in \{0,1\}$, and outputs ciphertext ct.

$\mathsf{Dec}(\mathsf{mpk}, f, \mathsf{sk}_f, x, \mathsf{ct}) \to \mu'$**.** The deterministic decryption algorithm takes as input master public key mpk, circuit $f \in \mathcal{F}_{\ell,d}$, secret key sk_f, attribute $x \in \{0,1\}^\ell$, ciphertext ct, and outputs $\mu' \in \{0,1\} \cup \{\bot\}$.

Definition 6.25 (Selectively Secure ABE with Succinct Keys)**.** A scheme $(\mathsf{Setup}, \mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$ is said to be a selectively secure ABE scheme with succinct keys for circuits if it satisfies the following properties.

Correctness. For any $\lambda, \ell, d \in \mathbb{N}$, $f \in \mathcal{F}_{\ell,d}$, $x \in \{0,1\}^\ell$ such that $f(x) = 1$, and $\mu \in \{0,1\}$,

$$\Pr\left[\mu = \mathsf{Dec}(\mathsf{mpk}, f, \mathsf{sk}_f, x, \mathsf{ct}) \;:\; \begin{array}{l}(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda, 1^\ell, 1^d),\\ \mathsf{sk}_f \leftarrow \mathsf{KGen}(\mathsf{msk}, f),\\ \mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{mpk}, x, \mu)\end{array}\right] = 1$$

Succinctness. There exist polynomials $\mathsf{poly}_{\mathsf{mpk}}(\cdot,\cdot,\cdot), \mathsf{poly}_{\mathsf{sk}}(\cdot,\cdot,\cdot), \mathsf{poly}_{\mathsf{ct}}(\cdot,\cdot,\cdot)$ such that $|\mathsf{mpk}| = \mathsf{poly}_{\mathsf{mpk}}(\lambda, \ell, d)$, $|\mathsf{sk}_f| = \mathsf{poly}_{\mathsf{sk}}(\lambda, \ell, d)$, and $|\mathsf{ct}| = \mathsf{poly}_{\mathsf{ct}}(\lambda, \ell, d)$ for any $\lambda, \ell, d \in \mathbb{N}$ and $f \in \mathcal{F}_{\ell,d}$. In particular, $|\mathsf{sk}_f|$ does not depend on the size of f.

Selective Security. For any stateful admissible PPT adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for any $\lambda \in \mathbb{N}$,

$$\left|\Pr\left[b \leftarrow \mathcal{A}^{\mathsf{KGen}(\mathsf{msk},\cdot)}(\mathsf{ct}^*) \;:\; \begin{array}{l}(1^\ell, 1^d, x^*) \leftarrow \mathcal{A}(1^\lambda),\\ (\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda, 1^\ell, 1^d),\\ (\mu_0, \mu_1) \leftarrow \mathcal{A}^{\mathsf{KGen}(\mathsf{msk},\cdot)}(\mathsf{mpk}),\\ b \leftarrow \{0,1\}, \mathsf{ct}^* \leftarrow \mathsf{Enc}(\mathsf{mpk}, x^*, \mu_b)\end{array}\right] - \frac{1}{2}\right| \leq \mathsf{negl}(\lambda)$$

A stateful PPT machine $\mathcal{A}$ is said to be admissible if $x^* \in \{0,1\}^\ell$ and every circuit f that it queries to $\mathsf{KGen}(\mathsf{msk}, \cdot)$ satisfies $f \in \mathcal{F}_{\ell,d}$ and $f(x^*) = 0$. This condition applies both before and after receiving ct^*.

Theorem 6.26 ([BGG+14])**.** Assuming the hardness of LWE with modulus-to-noise ratio $2^{\tilde{O}(d)}$, there exists a selectively secure ABE scheme with succinct keys for circuits of depth at most d (Definition 6.25).

Set Membership Circuits. For $N \in \mathbb{N}$ and $S \subseteq [N]$, let f_S denote the set membership circuit over $\lceil \log N \rceil$ input bits where $f_S(\mathsf{encvec}(i)) = 1$ if $i \in S$ and 0 otherwise. By Claim 4.10, f_S has depth at most $d_{N,B} := 1 + 2\lceil \log \lceil \log N \rceil \rceil + 2\lceil \log B \rceil$ for every $S \subseteq [N]$ of size at most B. So, $f_S \in \mathcal{F}_{\lceil \log N \rceil, d_{N,N}}$ for every $S \subseteq [N]$, where $d_{N,N} = O(\log N)$.

Construction 6.27. We construct a robust PPS scheme for $\mathcal{M}_\lambda = \{0,1\}$ using a selectively secure ABE scheme ABE with succinct keys (Definition 6.25) as follows.

$\mathsf{Setup}(1^\lambda, N)$**.** Sample $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{ABE.Setup}(1^\lambda, 1^{\lceil \log N \rceil}, 1^{d_{N,N}})$. Output $\mathsf{pp} := \mathsf{mpk}$ and msk.

$\mathsf{pkEnc}(i)$**.** Output $\mathsf{ct} \leftarrow \mathsf{ABE.Enc}(\mathsf{mpk}, \mathsf{encvec}(i), 0)$.

$\mathsf{KGen}(\mathsf{msk}, S)$**.** Output $\mathsf{sk}_S \leftarrow \mathsf{ABE.KGen}(\mathsf{msk}, f_S)$.

$\mathsf{skEnc}(\mathsf{msk}, i, \mu)$**.** Output $\mathsf{ct} \leftarrow \mathsf{ABE.Enc}(\mathsf{mpk}, \mathsf{encvec}(i), \mu)$.

$\mathsf{Decode}(S, \mathsf{sk}_S, i, \mathsf{ct})$**.** If $i \notin S$, output $\perp$. Otherwise, output $\mathsf{ABE.Dec}(\mathsf{mpk}, f_S, \mathsf{sk}_S, \mathsf{encvec}(i), \mathsf{ct})$.

Correctness. For $S \subseteq [N]$ and $i \in S$, $f_S \in \mathcal{F}_{\lceil \log N \rceil, d_{N,N}}$ and $f_S(\mathsf{encvec}(i)) = 1$. So, correctness holds by correctness of ABE.

Compactness. By succinctness of ABE with $\ell = \lceil \log N \rceil$ and $d = d_{N,N} = O(\log N)$, $|\mathsf{sk}_S| = \mathsf{poly}_{\mathsf{sk}}(\lambda, \lceil \log N \rceil, d_{N,N}) = \mathsf{poly}(\lambda, \log N)$, which does not depend on $|S|$, and $|\mathsf{ct}| = \mathsf{poly}(\lambda, \log N)$. Moreover, $|\mathsf{pp}| = |\mathsf{mpk}| = \mathsf{poly}(\lambda, \log N)$.

Theorem 6.28. If ABE is a selectively secure ABE scheme with succinct keys (Definition 6.25), then Construction 6.27 satisfies robust payload hiding (Definition 6.14).

Proof. This is similar to the proof of Theorem 6.23. The reduction adversary $\mathcal{B}$ receives (N, i^*, μ_{i^*}) from $\mathcal{A}$ and sends $(1^{\lceil \log N \rceil}, 1^{d_{N,N}}, \mathsf{encvec}(i^*))$ to the ABE challenger. It receives mpk, sends $(0, \mu_{i^*})$, and receives ct^*. It sends $(\mathsf{mpk}, \mathsf{ct}^*)$ to $\mathcal{A}$, answers (Enc, i, μ) with $\mathsf{ABE.Enc}(\mathsf{mpk}, \mathsf{encvec}(i), \mu)$, answers (KGen, S) by querying f_S to $\mathsf{KGen}(\mathsf{msk}, \cdot)$, and outputs the guess of $\mathcal{A}$. Every queried circuit f_S lies in $\mathcal{F}_{\lceil \log N \rceil, d_{N,N}}$ and satisfies $f_S(\mathsf{encvec}(i^*)) = 0$ as $i^* \notin S$. So, $\mathcal{B}$ is admissible and has the same advantage as $\mathcal{A}$. □

Instantiations. Plugging Theorem 6.26 into Construction 6.27 gives a robust PPS for $N = 2^\lambda$ from LWE with modulus-to-noise ratio $2^{\tilde{O}(\lambda)}$, as $d_{N,N} = O(\lambda)$. If we only need sets of size at most B, we run $\mathsf{ABE.Setup}$ with depth bound $d_{N,B}$ instead and obtain a robust PPS for sets of size at most B with compactness $\mathsf{poly}(\lambda, \log N, \log B)$ from LWE with modulus-to-noise ratio $2^{\tilde{O}(\log B)}$, which is quasi-polynomial for $B = \mathsf{poly}(\lambda)$. Plugging these into Construction 6.17 and using Theorem 6.28, Theorem 6.19, and Remark 6.18, we get the following.

Corollary 6.29. Assuming the hardness of LWE with sub-exponential modulus-to-noise ratio, there exists an adaptively secure CSS scheme for evolving graphs with unbounded collusion and unbounded degree where the shares have size $\mathsf{poly}(\lambda)$ and there is no public share. For any polynomial degree bound Δ, and in particular for graphs with $\mathsf{poly}(\lambda)$ vertices that are fixed in advance, LWE with quasi-polynomial modulus-to-noise ratio suffices.

Comparison with [ABI+23]. [ABI+23, Section 1.1.3] observe that ABE with security against multiple secret keys is equivalent to secret sharing for bipartite graphs where the vertices on one side receive secret keys and the vertices on the other side receive ciphertexts, and that [BGG+14] has succinct keys in the required sense. They only use this observation to construct secret sharing schemes for general access structures on n parties with share size $\tilde{O}(2^{n/2})$. Their CSS scheme for graphs from LWE goes through their PPRG and has share size $\sqrt{L} \cdot \mathsf{poly}(\lambda)$ for graphs with L vertices [ABI+23, Corollary 4.2], and their schemes with share size $\mathsf{poly}(\lambda, \log L)$ rely on RSA or on iO and SSB hash functions. Corollary 6.29 gives share size $\mathsf{poly}(\lambda, \log L)$ for every graph from LWE. The PPRGs from LWE in [ABI+23, HWW25] fix their output length at setup. Francati and Venturi [FV25] obtain evolving graph sharing from RSA or iO and SSB hashing. Our construction gives compact shares from LWE and supports adaptive graph growth and corruptions.

References

[ABI+23] Benny Applebaum, Amos Beimel, Yuval Ishai, Eyal Kushilevitz, Tianren Liu, and Vinod Vaikuntanathan. Succinct computational secret sharing. In Barna Saha and Rocco A. Servedio, editors, *55th Annual ACM Symposium on Theory of Computing*, pages 1553–1566, Orlando, FL, USA, June 20–23, 2023. ACM Press.

[ABN+22] Benny Applebaum, Amos Beimel, Oded Nir, Naty Peter, and Toniann Pitassi. Secret sharing, slice formulas, and monotone real circuits. In Mark Braverman, editor, *ITCS 2022: 13th Innovations in Theoretical Computer Science Conference*, volume 215, pages 8:1–8:23, Berkeley, CA, USA, January 31 – February 3, 2022. Leibniz International Proceedings in Informatics (LIPIcs).

[AFP25] Amit Agarwal, Rex Fernando, and Benny Pinkas. Efficiently-thresholdizable batched identity based encryption, with applications. In Yael Tauman Kalai and Seny F. Kamara, editors, *Advances in Cryptology – CRYPTO 2025, Part III*, volume 16002 of *Lecture Notes in Computer Science*, pages 69–100. Springer, Cham, Switzerland, August 2025.

[AJL+12] Gilad Asharov, Abhishek Jain, Adriana López-Alt, Eran Tromer, Vinod Vaikuntanathan, and Daniel Wichs. Multiparty computation with low communication, computation and interaction via threshold FHE. In David Pointcheval and Thomas Johansson, editors, *Advances in Cryptology – EUROCRYPT 2012*, volume 7237 of *Lecture Notes in Computer Science*, pages 483–501, Cambridge, UK, April 15–19, 2012. Springer Berlin Heidelberg, Germany.

[AMR25] Damiano Abram, Giulio Malavolta, and Lawrence Roy. Key-homomorphic computations for RAM: Fully succinct randomised encodings and more. In Yael Tauman Kalai and Seny F. Kamara, editors, *Advances in Cryptology – CRYPTO 2025, Part III*, volume 16002 of *Lecture Notes in Computer Science*, pages 236–268, Santa Barbara, CA, USA, August 17–21, 2025. Springer, Cham, Switzerland.

[AMR26] Damiano Abram, Giulio Malavolta, and Lawrence Roy. Tree encodings i: How to authenticate a truth table. Cryptology ePrint Archive, Paper 2026/741, 2026.

[AV19] Prabhanjan Ananth and Vinod Vaikuntanathan. Optimal bounded-collusion secure functional encryption. In *Theory of Cryptography Conference*, pages 174–198. Springer, 2019.

[BDH+26] Pedro Branco, Nico Döttling, Yao-Ching Hsieh, Abhishek Jain, Akshayaram Srinivasan, and Brent Waters. Hidden-bits generators with succinct CRS and openings. To appear at TCC 2026, 2026.

[BGG+14] Dan Boneh, Craig Gentry, Sergey Gorbunov, Shai Halevi, Valeria Nikolaenko, Gil Segev, Vinod Vaikuntanathan, and Dhinakaran Vinayagamurthy. Fully key-homomorphic encryption, arithmetic circuit abe and compact garbled circuits. In *Advances in Cryptology–EUROCRYPT 2014: 33rd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Copenhagen, Denmark, May 11-15, 2014. Proceedings 33*, pages 533–556. Springer, 2014.

[BL90] Josh Cohen Benaloh and Jerry Leichter. Generalized secret sharing and monotone functions. In Shafi Goldwasser, editor, *Advances in Cryptology – CRYPTO'88*, volume 403 of *Lecture Notes in Computer Science*, pages 27–35, Santa Barbara, CA, USA, August 21–25, 1990. Springer, New York, USA.

[BLSV18] Zvika Brakerski, Alex Lombardi, Gil Segev, and Vinod Vaikuntanathan. Anonymous IBE, leakage resilience and circular security from new assumptions. In Jesper Buus Nielsen and Vincent Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2018, Part I*, volume 10820 of *Lecture Notes in Computer Science*, pages 535–564, Tel Aviv, Israel, April 29 – May 3, 2018. Springer, Cham, Switzerland.

[BR93] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *Proceedings of the 1st ACM Conference on Computer and Communications Security*, pages 62–73, 1993.

[BSW11] Dan Boneh, Amit Sahai, and Brent Waters. Functional encryption: Definitions and challenges. In *Theory of Cryptography: 8th Theory of Cryptography Conference, TCC 2011, Providence, RI, USA, March 28-30, 2011. Proceedings 8*, pages 253–273. Springer, 2011.

[BZ14] Dan Boneh and Mark Zhandry. Multiparty key exchange, efficient traitor tracing, and more from indistinguishability obfuscation. In *34rd Annual International Cryptology Conference, CRYPTO 2014*, pages 480–499. Springer Verlag, 2014.

[CHW25] Jeffrey Champion, Yao-Ching Hsieh, and David J. Wu. Registered ABE and adaptively-secure broadcast encryption from succinct LWE. In Yael Tauman Kalai and Seny F. Kamara, editors, *Advances in Cryptology – CRYPTO 2025, Part III*, volume 16002 of *Lecture Notes in Computer Science*, pages 3–34, Santa Barbara, CA, USA, August 17–21, 2025. Springer, Cham, Switzerland.

[Csi95] László Csirmaz. The size of a share must be large. In Alfredo De Santis, editor, *Advances in Cryptology – EUROCRYPT'94*, volume 950 of *Lecture Notes in Computer Science*, pages 13–22, Perugia, Italy, May 9–12, 1995. Springer Berlin Heidelberg, Germany.

[Csi96] László Csirmaz. The dealer's random bits in perfect secret sharing schemes. *Studia Scientiarum Mathematicarum Hungarica*, 32(3):429–438, 1996.

[CW24] Jeffrey Champion and David J. Wu. Distributed broadcast encryption from lattices. In Elette Boyle and Mohammad Mahmoody, editors, *TCC 2024: 22nd Theory of Cryptography Conference, Part III*, volume 15366 of *Lecture Notes in Computer Science*, pages 156–189, Milan, Italy, December 2–6, 2024. Springer, Cham, Switzerland.

[CW26] Jeffrey Champion and David J. Wu. Optimal distributed monotone-policy encryption for DNFs and more from lattices. Cryptology ePrint Archive, Paper 2026/1464, 2026.

[FGV25] Danilo Francati, Sara Giammusso, and Daniele Venturi. Taming adaptive security and new access structures in evolving secret sharing. In Goichiro Hanaoka and Bo-Yin Yang, editors, *Advances in Cryptology – ASIACRYPT 2025, Part VIII*, volume 16252 of *Lecture Notes in Computer Science*, pages 67–98, Melbourne, VIC, Australia, December 8–12, 2025. Springer, Singapore, Singapore.

[FJP15] Georg Fuchsbauer, Zahra Jafargholi, and Krzysztof Pietrzak. A quasipolynomial reduction for generalized selective decryption on trees. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *Advances in Cryptology – CRYPTO 2015, Part I*, volume 9215 of *Lecture Notes in Computer Science*, pages 601–620, Santa Barbara, CA, USA, August 16–20, 2015. Springer Berlin Heidelberg, Germany.

[FKPR14] Georg Fuchsbauer, Momchil Konstantinov, Krzysztof Pietrzak, and Vanishree Rao. Adaptive security of constrained PRFs. In Palash Sarkar and Tetsu Iwata, editors, *Advances in Cryptology – ASIACRYPT 2014, Part II*, volume 8874 of *Lecture Notes in Computer Science*, pages 82–101, Kaoshiung, Taiwan, R.O.C., December 7–11, 2014. Springer Berlin Heidelberg, Germany.

[FN94] Amos Fiat and Moni Naor. Broadcast encryption. In Douglas R. Stinson, editor, *Advances in Cryptology – CRYPTO'93*, volume 773 of *Lecture Notes in Computer Science*, pages 480–491, Santa Barbara, CA, USA, August 22–26, 1994. Springer Berlin Heidelberg, Germany.

[FV25] Danilo Francati and Daniele Venturi. Evolving secret sharing revisited: computational security and succinctness. *Designs, Codes and Cryptography*, 93(9):3809–3861, 2025.

[FWW23] Cody Freitag, Brent Waters, and David J. Wu. How to use (plain) witness encryption: Registered ABE, flexible broadcast, and more. In Helena Handschuh and Anna Lysyanskaya, editors, *Advances in Cryptology – CRYPTO 2023, Part IV*, volume 14084 of *Lecture Notes in Computer Science*, pages 498–531, Santa Barbara, CA, USA, August 20–24, 2023. Springer, Cham, Switzerland.

[GKW18] Rishab Goyal, Venkata Koppula, and Brent Waters. Collusion resistant traitor tracing from learning with errors. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*, pages 660–670, 2018.

[GLWW23] Rachit Garg, George Lu, Brent Waters, and David J. Wu. Realizing flexible broadcast encryption: How to broadcast to a public-key directory. In Weizhi Meng,

Christian Damsgaard Jensen, Cas Cremers, and Engin Kirda, editors, *ACM CCS 2023: 30th Conference on Computer and Communications Security*, pages 1093–1107, Copenhagen, Denmark, November 26–30, 2023. ACM Press.

[GPSW06] Vipul Goyal, Omkant Pandey, Amit Sahai, and Brent Waters. Attribute-based encryption for fine-grained access control of encrypted data. In *Proceedings of the 13th ACM conference on Computer and communications security*, pages 89–98, 2006.

[GSW13] Craig Gentry, Amit Sahai, and Brent Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In *Advances in Cryptology–CRYPTO 2013: 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part I*, pages 75–92. Springer, 2013.

[GVW12] Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Functional encryption with bounded collusions via multi-party computation. In *Advances in Cryptology–CRYPTO 2012: 32nd Annual Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2012. Proceedings*, pages 162–179. Springer, 2012.

[GW09] Craig Gentry and Brent Waters. Adaptive security in broadcast encryption systems (with short ciphertexts). In Antoine Joux, editor, *Advances in Cryptology – EUROCRYPT 2009*, volume 5479 of *Lecture Notes in Computer Science*, pages 171–188, Cologne, Germany, April 26–30, 2009. Springer Berlin Heidelberg, Germany.

[GWWW26] Junqing Gong, Brent Waters, Hoeteck Wee, and David J. Wu. Threshold batched identity-based encryption from pairings in the plain model. In *Advances in Cryptology – EUROCRYPT 2026*, Lecture Notes in Computer Science. Springer, Cham, Switzerland, 2026. Cryptology ePrint Archive, Paper 2025/2103.

[GY26a] Rishab Goyal and Saikumar Yadugiri. Adaptively-secure flexible and identity-based broadcast encryption from lattices. Cryptology ePrint Archive, Paper 2026/862, 2026.

[GY26b] Rishab Goyal and Saikumar Yadugiri. Distributed monotone policy encryption with stronger security for DNFs and threshold policies from lattices. Cryptology ePrint Archive, Paper 2026/1609, 2026.

[GY26c] Rishab Goyal and Saikumar Yadugiri. Equivocal broadcast encryption: Adaptively-secure optimal distributed broadcast encryption from lattices. In Nadia Heninger and Mike Rosulek, editors, *Advances in Cryptology – CRYPTO 2026*, pages 414–445, Cham, 2026. Springer Nature Switzerland.

[HJO+16] Brett Hemenway, Zahra Jafargholi, Rafail Ostrovsky, Alessandra Scafuro, and Daniel Wichs. Adaptively secure garbled circuits from one-way functions. In Matthew Robshaw and Jonathan Katz, editors, *Advances in Cryptology – CRYPTO 2016, Part III*, volume 9816 of *Lecture Notes in Computer Science*, pages 149–178, Santa Barbara, CA, USA, August 14–18, 2016. Springer Berlin Heidelberg, Germany.

[HJW26] Yao-Ching Hsieh, Abhishek Jain, and Brent Waters. New applications of RSA. Cryptology ePrint Archive, Paper 2026/2167, 2026.

[HW09] Susan Hohenberger and Brent Waters. Short and stateless signatures from the RSA assumption. In Shai Halevi, editor, *Advances in Cryptology – CRYPTO 2009*, volume 5677 of *Lecture Notes in Computer Science*, pages 654–670, Santa Barbara, CA, USA, August 16–20, 2009. Springer Berlin Heidelberg, Germany.

[HWW25] Yao-Ching Hsieh, Brent Waters, and David J. Wu. A generic approach to adaptively-secure broadcast encryption in the plain model. In Serge Fehr and Pierre-Alain Fouque, editors, *Advances in Cryptology – EUROCRYPT 2025, Part III*, volume 15603 of *Lecture Notes in Computer Science*, pages 336–365, Madrid, Spain, May 4–8, 2025. Springer, Cham, Switzerland.

[JKK+17] Zahra Jafargholi, Chethan Kamath, Karen Klein, Ilan Komargodski, Krzysztof Pietrzak, and Daniel Wichs. Be adaptive, avoid overcommitting. In Jonathan Katz and Hovav Shacham, editors, *Advances in Cryptology – CRYPTO 2017, Part I*, volume 10401 of *Lecture Notes in Computer Science*, pages 133–163, Santa Barbara, CA, USA, August 20–24, 2017. Springer, Cham, Switzerland.

[JW16] Zahra Jafargholi and Daniel Wichs. Adaptive security of Yao's garbled circuits. In Martin Hirt and Adam D. Smith, editors, *TCC 2016-B: 14th Theory of Cryptography Conference, Part I*, volume 9985 of *Lecture Notes in Computer Science*, pages 433–458, Beijing, China, October 31 – November 3, 2016. Springer Berlin Heidelberg, Germany.

[KNY16] Ilan Komargodski, Moni Naor, and Eylon Yogev. How to share a secret, infinitely. In Martin Hirt and Adam D. Smith, editors, *TCC 2016-B: 14th Theory of Cryptography Conference, Part II*, volume 9986 of *Lecture Notes in Computer Science*, pages 485–514, Beijing, China, October 31 – November 3, 2016. Springer Berlin Heidelberg, Germany.

[KW93] Mauricio Karchmer and Avi Wigderson. On span programs. In *Proceedings of Structures in Complexity Theory*, pages 102–111, 1993.

[KW20] Lucas Kowalczyk and Hoeteck Wee. Compact adaptively secure ABE for NC^1 from k-Lin. *Journal of Cryptology*, 33(3):954–1002, July 2020.

[LV18] Tianren Liu and Vinod Vaikuntanathan. Breaking the circuit-size barrier in secret sharing. In Ilias Diakonikolas, David Kempe, and Monika Henzinger, editors, *50th Annual ACM Symposium on Theory of Computing*, pages 699–708, Los Angeles, CA, USA, June 25–29, 2018. ACM Press.

[LVW18] Tianren Liu, Vinod Vaikuntanathan, and Hoeteck Wee. Towards breaking the exponential barrier for general secret sharing. In Jesper Buus Nielsen and Vincent Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2018, Part I*, volume 10820 of *Lecture Notes in Computer Science*, pages 567–596, Tel Aviv, Israel, April 29 – May 3, 2018. Springer, Cham, Switzerland.

[LW25] George Lu and Brent Waters. How to make any computational secret sharing scheme adaptively secure. In Yael Tauman Kalai and Seny F. Kamara, editors, *Advances in Cryptology – CRYPTO 2025, Part IV*, volume 16003 of *Lecture Notes in Computer*

Science, pages 328–355, Santa Barbara, CA, USA, August 17–21, 2025. Springer, Cham, Switzerland.

[Lyu12] Vadim Lyubashevsky. Lattice signatures without trapdoors. In David Pointcheval and Thomas Johansson, editors, *Advances in Cryptology – EUROCRYPT 2012*, volume 7237 of *Lecture Notes in Computer Science*, pages 738–755, Cambridge, UK, April 15–19, 2012. Springer Berlin Heidelberg, Germany.

[MP12] Daniele Micciancio and Chris Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. In David Pointcheval and Thomas Johansson, editors, *Advances in Cryptology – EUROCRYPT 2012*, volume 7237 of *Lecture Notes in Computer Science*, pages 700–718, Cambridge, UK, April 15–19, 2012. Springer Berlin Heidelberg, Germany.

[Reg05] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In Harold N. Gabow and Ronald Fagin, editors, *37th Annual ACM Symposium on Theory of Computing*, pages 84–93, Baltimore, MA, USA, May 22–24, 2005. ACM Press.

[Sha79] Adi Shamir. How to share a secret. *Communications of the Association for Computing Machinery*, 22(11):612–613, November 1979.

[SW05] Amit Sahai and Brent Waters. Fuzzy identity-based encryption. In *Advances in Cryptology–EUROCRYPT 2005: 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Aarhus, Denmark, May 22-26, 2005. Proceedings 24*, pages 457–473. Springer, 2005.

[Wee24] Hoeteck Wee. Circuit ABE with $\mathsf{poly}(\mathsf{depth}, \lambda)$-sized ciphertexts and keys from lattices. In Leonid Reyzin and Douglas Stebila, editors, *Advances in Cryptology – CRYPTO 2024, Part III*, volume 14922 of *Lecture Notes in Computer Science*, pages 178–209, Santa Barbara, CA, USA, August 18–22, 2024. Springer, Cham, Switzerland.

[Wee25] Hoeteck Wee. Almost optimal KP and CP-ABE for circuits from succinct LWE. In Serge Fehr and Pierre-Alain Fouque, editors, *Advances in Cryptology – EUROCRYPT 2025, Part III*, volume 15603 of *Lecture Notes in Computer Science*, pages 34–62, Madrid, Spain, May 4–8, 2025. Springer, Cham, Switzerland.

[WQZD10] Qianhong Wu, Bo Qin, Lei Zhang, and Josep Domingo-Ferrer. Ad hoc broadcast encryption (poster presentation). In Ehab Al-Shaer, Angelos D. Keromytis, and Vitaly Shmatikov, editors, *ACM CCS 2010: 17th Conference on Computer and Communications Security*, pages 741–743, Chicago, Illinois, USA, October 4–8, 2010. ACM Press.

[WW24] Brent Waters and Daniel Wichs. Adaptively secure attribute-based encryption from witness encryption. In Elette Boyle and Mohammad Mahmoody, editors, *TCC 2024: 22nd Theory of Cryptography Conference, Part III*, volume 15366 of *Lecture Notes in Computer Science*, pages 65–90, Milan, Italy, December 2–6, 2024. Springer, Cham, Switzerland.

[WW25] Hoeteck Wee and David J. Wu. Unbounded distributed broadcast encryption and registered ABE from succinct LWE. In Yael Tauman Kalai and Seny F. Kamara, editors, *Advances in Cryptology – CRYPTO 2025, Part III*, volume 16002 of *Lecture Notes in Computer Science*, pages 204–235, Santa Barbara, CA, USA, August 17–21, 2025. Springer, Cham, Switzerland.

[XC25] Saisi Xiong and Jie Chen. Efficient batched IBE from lattices in the standard model. Cryptology ePrint Archive, Paper 2025/2158, 2025.

A Q-Bounded Robust PPS from PPS

A PPS (Definition 4.1) only guarantees payload hiding against a single secret key. In this section, we show that PPS gives a robust PPS against any polynomial number Q of secret keys fixed at setup, which we call a Q-bounded robust PPS, at the cost of ciphertexts and public parameters growing linearly with Q. Plugging it into Construction 6.17 gives a CSS scheme for evolving graphs with collusion bound Q from PPS alone. The construction has a locality property, where decoding reads only a small part of the public parameters, which lets the dealer avoid a public share altogether.

Definition A.1 (Q-Bounded Robust PPS)**.** An RPPS scheme $(\mathsf{Setup}, \mathsf{pkEnc}, \mathsf{KGen}, \mathsf{skEnc}, \mathsf{Decode})$, where $\mathsf{Setup}(1^\lambda, N, 1^Q)$ additionally takes a bound Q on the number of secret keys, is said to be a Q-bounded robust PPS scheme if it satisfies the correctness and compactness properties of Definition 6.14 where $|\mathsf{ct}|$ may additionally grow polynomially with Q, and the following bounded adaptive robustness property.

Bounded Adaptive Robustness. For any stateful admissible PPT adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for any $\lambda \in \mathbb{N}$,

$$\left| \Pr \left[b \leftarrow \mathcal{A}^{\mathcal{O}(\cdot,\cdot)}(\mathsf{pp}, \mathsf{ct}^*) \quad : \quad \begin{array}{l} (N, 1^Q, i^* \in [N], \mu_0, \mu_1 \in \mathcal{M}_\lambda) \leftarrow \mathcal{A}(1^\lambda), \\ b \leftarrow \{0,1\}, (\mathsf{pp}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda, N, 1^Q), \\ \mathsf{ct}^* \leftarrow \mathsf{skEnc}(\mathsf{msk}, i^*, \mu_b) \end{array} \right] - \frac{1}{2} \right| \leq \mathsf{negl}(\lambda)$$

where $\mathcal{O}$ is a stateful oracle that responds to the following queries.

(Enc, i, μ)**.** Given $i \in [N]$ and $\mu \in \mathcal{M}_\lambda$, $\mathcal{O}$ replies with $\mathsf{ct} \leftarrow \mathsf{skEnc}(\mathsf{msk}, i, \mu)$.

(KGen, S)**.** Given $S \subseteq [N]$, $\mathcal{O}$ replies with $\mathsf{sk}_S \leftarrow \mathsf{KGen}(\mathsf{msk}, S)$.

A stateful PPT machine $\mathcal{A}$ is said to be admissible if every query (Enc, i, μ) satisfies $i \neq i^*$, every query (KGen, S) satisfies $i^* \notin S$, and it makes at most Q queries of the form $(\mathsf{KGen}, \cdot)$.

We use the following preliminaries in the construction.

Shamir Secret Sharing. Let $\mathbb{F}$ be a finite field and $\ell < |\mathbb{F}|$. Fix distinct non-zero elements $\alpha_1, \ldots, \alpha_\ell \in \mathbb{F}$. Shamir secret sharing scheme [Sha79] over $\mathbb{F}$ consists of the following polynomial time algorithms.

$\mathsf{Share}(1^\ell, 1^t, s) \to (\mathsf{sh}_1, \ldots, \mathsf{sh}_\ell)$**.** The randomized sharing algorithm takes as input number of shares ℓ, threshold $t \leq \ell$, secret $s \in \mathbb{F}$, and outputs the shares sh_u for each $u \in [\ell]$.

$\mathsf{Recon}(\Gamma, (\mathsf{sh}_u)_{u\in\Gamma}) \to s'$**.** The deterministic reconstruction algorithm takes as input a set $\Gamma \subseteq [\ell]$, shares $(\mathsf{sh}_u)_{u\in\Gamma}$, and outputs s'.

Lemma A.2 ([Sha79])**.** For any $\ell < |\mathbb{F}|$, $t \le \ell$, and $s \in \mathbb{F}$, the following properties hold for $(\mathsf{sh}_1, \ldots, \mathsf{sh}_\ell) \leftarrow \mathsf{Share}(1^\ell, 1^t, s)$.

Correctness. For any $\Gamma \subseteq [\ell]$ with $|\Gamma| \ge t$, $\mathsf{Recon}(\Gamma, (\mathsf{sh}_u)_{u\in\Gamma}) = s$.

Privacy. For any $\Gamma \subseteq [\ell]$ with $|\Gamma| < t$, $(\mathsf{sh}_u)_{u\in\Gamma}$ is uniformly random in $\mathbb{F}^{|\Gamma|}$. In particular, it is independent of s.

Lemma A.3 (Small Pairwise Intersections)**.** Let $\lambda \in \mathbb{N}$. Define $R := \lambda^2(\lambda+1)^2$. For any $c' \le \lambda$, let $\Gamma_1, \ldots, \Gamma_{c'} \subseteq [R]$ be independent and uniformly random subsets of size $\lambda+1$. Then,

$$\Pr\left[\left|\bigcup_{j\neq j'} (\Gamma_j \cap \Gamma_{j'})\right| \ge \lambda+1\right] \le \mathsf{negl}(\lambda)$$

Proof. This is a variant of [GVW12, Lemma B.1] for our parameters. We give a proof for completeness. Let $B := \bigcup_{j\neq j'}(\Gamma_j \cap \Gamma_{j'})$. If $|B| \ge \lambda+1$, then there exist distinct $u_1, \ldots, u_{\lambda+1} \in B$ and pairs $j_k \neq j'_k$ such that $u_k \in \Gamma_{j_k} \cap \Gamma_{j'_k}$ for each $k \in [\lambda+1]$. Fix such elements and pairs. Then, each Γ_j must contain some d_j fixed distinct elements where $\sum_{j\in[c']} d_j = 2(\lambda+1)$. As Γ_j is a uniformly random subset of $[R]$ of size $\lambda+1$, this happens with probability at most $((\lambda+1)/R)^{d_j}$. As $\Gamma_1, \ldots, \Gamma_{c'}$ are independent, all these memberships hold with probability at most $((\lambda+1)/R)^{2(\lambda+1)}$.

There are at most $\binom{R}{\lambda+1} \le (eR/(\lambda+1))^{\lambda+1}$ choices for the elements and at most $\binom{c'}{2}^{\lambda+1} \le (\lambda^2/2)^{\lambda+1}$ choices for the pairs as $c' \le \lambda$. So, by a union bound, $\Pr[|B| \ge \lambda+1] \le (e\lambda^2(\lambda+1)/2R)^{\lambda+1} = (e/2(\lambda+1))^{\lambda+1}$ as $R = \lambda^2(\lambda+1)^2$. This is negligible in λ. Thus, the lemma holds. □

Lemma A.4 (Load Balancing [AV19], adapted)**.** Let $\lambda \in \mathbb{N}$ and $Q = \mathsf{poly}(\lambda)$. Let $k_1, \ldots, k_Q \leftarrow [Q]$ be independent and uniformly random. Then, $\Pr[\exists\, k \in [Q] \text{ such that } |\{j \in [Q] : k_j = k\}| > \lambda] \le \mathsf{negl}(\lambda)$.

Proof. For each $k \in [Q]$, the number of $j \in [Q]$ with $k_j = k$ is a sum of Q independent indicator random variables with expectation 1. By a Chernoff bound, it is greater than λ with probability at most $e^{-(\lambda-1)^2/(\lambda+1)}$. The lemma follows by a union bound over $k \in [Q]$ [AV19, Section 4]. □

We now proceed with the construction.

Notation and Parameters. Let $\lambda \in \mathbb{N}$ be the security parameter. Define $R := \lambda^2(\lambda+1)^2$ and $h := \lceil \log(R+1) \rceil$. We use Shamir secret sharing over the finite field $\mathbb{F}$ of size 2^h ($> R$) and identify $0, 1 \in \mathbb{F}$ with the bits $0, 1$, so that $\{0,1\} \subseteq \mathbb{F}$.

Construction A.5. We construct a Q-bounded robust PPS scheme (Definition A.1) for $\mathcal{M}_\lambda = \mathbb{F}$ using a PPS scheme PPS for $\mathcal{M}_\lambda = \mathbb{F}$ (Definition 4.1) as follows. For each $k \in [Q]$ and $r \in [R]$, the algorithms of the (k,r)-th PPS instance take $\mathsf{pps.crs}_{k,r}$ implicitly.

$\mathsf{Setup}(1^\lambda, N, 1^Q)$**.** For each $k \in [Q]$ and $r \in [R]$, sample $\mathsf{pps.crs}_{k,r} \leftarrow \mathsf{PPS.GSetup}(1^\lambda, N)$ and $\mathsf{pps.msk}_{k,r} \leftarrow \mathsf{PPS.MSKGen}(\mathsf{pps.crs}_{k,r})$.

Output $\mathsf{pp} := (\mathsf{pps.crs}_{k,r})_{k\in[Q],r\in[R]}$ and $\mathsf{msk} := (\mathsf{pps.msk}_{k,r})_{k\in[Q],r\in[R]}$.

$\mathsf{pkEnc}(i)$**.** For each $k \in [Q]$ and $r \in [R]$, sample $\mathsf{pps.ct}_{k,r} \leftarrow \mathsf{PPS.pkEnc}(i)$.

Output $\mathsf{ct} := (\mathsf{pps.ct}_{k,r})_{k\in[Q],r\in[R]}$.

$\mathsf{KGen}(\mathsf{msk}, S)$**.** Parse msk as $(\mathsf{pps.msk}_{k,r})_{k,r}$. Sample $k \leftarrow [Q]$ and a uniformly random set $\Gamma \subseteq [R]$ of size $\lambda + 1$. For each $r \in \Gamma$, sample $\mathsf{pps.sk}_{S,r} \leftarrow \mathsf{PPS.KGen}(\mathsf{pps.msk}_{k,r}, S)$.

Output $\mathsf{sk}_S := (k, \Gamma, (\mathsf{pps.crs}_{k,r}, \mathsf{pps.sk}_{S,r})_{r\in\Gamma})$.

$\mathsf{skEnc}(\mathsf{msk}, i, \mu)$**.** Parse msk as $(\mathsf{pps.msk}_{k,r})_{k,r}$. For each $k \in [Q]$, sample $(\sigma_{k,1}, \ldots, \sigma_{k,R}) \leftarrow \mathsf{Shamir.Share}(1^R, 1^{\lambda+1}, \mu)$ and for each $r \in [R]$, sample $\mathsf{pps.ct}_{k,r} \leftarrow \mathsf{PPS.skEnc}(\mathsf{pps.msk}_{k,r}, i, \sigma_{k,r})$.

Output $\mathsf{ct} := (\mathsf{pps.ct}_{k,r})_{k\in[Q],r\in[R]}$.

$\mathsf{Decode}(S, \mathsf{sk}_S, i, \mathsf{ct})$**.** If $i \notin S$, output $\perp$. Otherwise, parse sk_S as $(k, \Gamma, (\mathsf{pps.crs}_{k,r}, \mathsf{pps.sk}_{S,r})_{r\in\Gamma})$ and ct as $(\mathsf{pps.ct}_{k,r})_{k,r}$. For each $r \in \Gamma$, compute $y_r = \mathsf{PPS.Decode}(S, \mathsf{pps.sk}_{S,r}, i, \mathsf{pps.ct}_{k,r})$ using $\mathsf{pps.crs}_{k,r}$. If $y_r = \perp$ for some $r \in \Gamma$, output $\perp$.

Output $\mu' := \mathsf{Shamir.Recon}(\Gamma, (y_r)_{r\in\Gamma})$.

Correctness. For each $r \in \Gamma$, $\mathsf{pps.ct}_{k,r}$ is a PPS ciphertext for index i and message $\sigma_{k,r}$ under $\mathsf{pps.msk}_{k,r}$, and $\mathsf{pps.sk}_{S,r}$ is a PPS secret key for S under the same master secret key where $i \in S$. So, by correctness of PPS, $y_r = \sigma_{k,r}$ for each $r \in \Gamma$. As $|\Gamma| = \lambda + 1$ and $(\sigma_{k,1}, \ldots, \sigma_{k,R}) \leftarrow \mathsf{Shamir.Share}(1^R, 1^{\lambda+1}, \mu)$, by correctness of Shamir secret sharing, $\mu' = \mu$. This holds for every bucket k and every Γ as μ is shared in every bucket.

Compactness. A secret key consists of $k \in [Q]$, the set Γ which we describe using its $\lambda + 1$ elements of $[R]$, and $\lambda + 1$ PPS common random strings and secret keys. By full compactness of PPS, $|\mathsf{sk}_S| = \lceil \log Q \rceil + (\lambda+1)\lceil \log R \rceil + (\lambda+1)(\mathsf{poly}_{\mathsf{crs}}(\lambda, \log N) + \mathsf{poly}_{\mathsf{sk}}(\lambda, \log N)) = \mathsf{poly}(\lambda, \log N)$ as $Q = \mathsf{poly}(\lambda)$ and $R = O(\lambda^4)$. Moreover, $|\mathsf{ct}| = QR \cdot \mathsf{poly}_{\mathsf{ct}}(\lambda, \log N) = Q \cdot \mathsf{poly}(\lambda, \log N)$ and $|\mathsf{pp}| = QR \cdot \mathsf{poly}_{\mathsf{crs}}(\lambda, \log N) = Q \cdot \mathsf{poly}(\lambda, \log N)$.

Locality. Decode does not read pp at all, as the secret key carries the $\lambda + 1$ common random strings it needs, and it reads only the $\lambda + 1$ components $\mathsf{pps.ct}_{k,r}$ for $r \in \Gamma$ of the ciphertext. So, with random access to ct, Decode reads $\mathsf{poly}(\lambda, \log N)$ bits, which is independent of $|S|$ and Q. When PPS is instantiated with Construction 4.6, the running time of Decode is $|S| \cdot \mathsf{poly}(\lambda, \log N)$, where the dependence on $|S|$ comes only from the set membership circuit f_S. In the secret sharing application, this means that the public share pp is never needed, as we discuss below. This is not the case for the LWE based PPRGs of [ABI+23, HWW25], where recovering the i-th output combines the projected key with the public values of every index in S.

A.1 Bounded Adaptive Robustness Analysis

Theorem A.6. If PPS satisfies payload hiding property (Definition 4.1) against post-key ciphertext queries (Remark 6.15), then Construction A.5 satisfies bounded adaptive robustness property (Definition A.1).

Proof. We prove this theorem using the following hybrids for an admissible adversary $\mathcal{A}$. Encryption and key queries of $\mathcal{A}$ may be interleaved arbitrarily. For each $v \in [0, QR]$, let P_v denote the set of the first v pairs of $[Q] \times [R]$ in lexicographic order.

$\mathbf{Hyb}_0^{\mathcal{A}}(1^\lambda)$. This is the bounded adaptive robustness experiment from Definition A.1 with $b = 0$.

Setup Phase. $\mathcal{A}$ sends $(N, 1^Q, i^*, \mu_0, \mu_1)$. Sample $\mathsf{pps.crs}_{k,r} \leftarrow \mathsf{PPS.GSetup}(1^\lambda, N)$ and $\mathsf{pps.msk}_{k,r} \leftarrow \mathsf{PPS.MSKGen}(\mathsf{pps.crs}_{k,r})$ for each $k \in [Q]$ and $r \in [R]$. For each $k \in [Q]$, sample $(\sigma^*_{k,1}, \ldots, \sigma^*_{k,R}) \leftarrow \mathsf{Shamir.Share}(1^R, 1^{\lambda+1}, \mu_0)$ and for each $r \in [R]$, sample $\mathsf{pps.ct}^*_{k,r} \leftarrow \mathsf{PPS.skEnc}(\mathsf{pps.msk}_{k,r}, i^*, \sigma^*_{k,r})$. Set $\mathsf{pp} := (\mathsf{pps.crs}_{k,r})_{k,r}$, $\mathsf{msk} := (\mathsf{pps.msk}_{k,r})_{k,r}$, and $\mathsf{ct}^* := (\mathsf{pps.ct}^*_{k,r})_{k,r}$. Send $(\mathsf{pp}, \mathsf{ct}^*)$ to $\mathcal{A}$.

Encryption Queries. Given (Enc, i, μ) with $i \neq i^*$, for each $k \in [Q]$, sample $(\sigma_{k,1}, \ldots, \sigma_{k,R}) \leftarrow \mathsf{Shamir.Share}(1^R, 1^{\lambda+1}, \mu)$ and for each $r \in [R]$, sample $\mathsf{pps.ct}_{k,r} \leftarrow \mathsf{PPS.skEnc}(\mathsf{pps.msk}_{k,r}, i, \sigma_{k,r})$. Send $\mathsf{ct} := (\mathsf{pps.ct}_{k,r})_{k,r}$ to $\mathcal{A}$.

Key Queries. Given the t-th query (KGen, S) with $i^* \notin S$, sample $k_t \leftarrow [Q]$ and $\Gamma_t \subseteq [R]$ of size $\lambda + 1$. For each $r \in \Gamma_t$, sample $\mathsf{pps.sk}_{S,r} \leftarrow \mathsf{PPS.KGen}(\mathsf{pps.msk}_{k_t,r}, S)$. Send $\mathsf{sk}_S := (k_t, \Gamma_t, (\mathsf{pps.crs}_{k_t,r}, \mathsf{pps.sk}_{S,r})_{r \in \Gamma_t})$ to $\mathcal{A}$.

Guess Phase. $\mathcal{A}$ outputs its guess b'. Output b'.

$\mathbf{Hyb}_1^{\mathcal{A}}(1^\lambda)$. This is the same as $\mathbf{Hyb}_0^{\mathcal{A}}(1^\lambda)$ except that we sample $k_t \leftarrow [Q]$ and $\Gamma_t \subseteq [R]$ of size $\lambda + 1$ for each $t \in [Q]$ during the setup phase and use (k_t, Γ_t) for the t-th key query. For each $k \in [Q]$, let $B_k := \bigcup_{t \neq t', k_t = k_{t'} = k} (\Gamma_t \cap \Gamma_{t'})$. If $|\{t \in [Q] : k_t = k\}| > \lambda$ or $|B_k| \geq \lambda + 1$ for some $k \in [Q]$, the challenger aborts and outputs 0. Note that for any $k \in [Q]$ and $r \in [R] \setminus B_k$, at most one key query uses $\mathsf{pps.msk}_{k,r}$.

$\mathbf{Hyb}_{2,v}^{\mathcal{A}}(1^\lambda)$ **for** $v \in [0, QR]$. This is the same as $\mathbf{Hyb}_1^{\mathcal{A}}(1^\lambda)$ except that for each $(k, r) \in P_v$ with $r \notin B_k$, we sample $\mathsf{pps.ct}^*_{k,r} \leftarrow \mathsf{PPS.pkEnc}(i^*)$ during the setup phase. Note that $\mathbf{Hyb}_{2,0}^{\mathcal{A}}(1^\lambda)$ is the same as $\mathbf{Hyb}_1^{\mathcal{A}}(1^\lambda)$.

$\mathbf{Hyb}_3^{\mathcal{A}}(1^\lambda)$. This is the same as $\mathbf{Hyb}_{2,QR}^{\mathcal{A}}(1^\lambda)$ except that for each $k \in [Q]$, we sample $(\sigma^*_{k,1}, \ldots, \sigma^*_{k,R}) \leftarrow \mathsf{Shamir.Share}(1^R, 1^{\lambda+1}, \mu_1)$ during the setup phase.

$\mathbf{Hyb}_{4,v}^{\mathcal{A}}(1^\lambda)$ **for** $v \in [0, QR]$. This is the same as $\mathbf{Hyb}_3^{\mathcal{A}}(1^\lambda)$ except that for each $(k, r) \in P_v$ with $r \notin B_k$, we sample $\mathsf{pps.ct}^*_{k,r} \leftarrow \mathsf{PPS.skEnc}(\mathsf{pps.msk}_{k,r}, i^*, \sigma^*_{k,r})$ during the setup phase where $\sigma^*_{k,r}$ is the share of μ_1. Note that $\mathbf{Hyb}_{4,0}^{\mathcal{A}}(1^\lambda)$ is the same as $\mathbf{Hyb}_3^{\mathcal{A}}(1^\lambda)$.

$\mathbf{Hyb}_5^{\mathcal{A}}(1^\lambda)$. This is the same as $\mathbf{Hyb}_{4,QR}^{\mathcal{A}}(1^\lambda)$ except that we sample $k_t \leftarrow [Q]$ and $\Gamma_t \subseteq [R]$ of size $\lambda + 1$ during the t-th key query and the challenger no longer aborts. Note that this is the bounded adaptive robustness experiment from Definition A.1 with $b = 1$.

Lemma A.7. For any admissible adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for any $\lambda \in \mathbb{N}$, $|\Pr[1 \leftarrow \mathbf{Hyb}_0^{\mathcal{A}}(1^\lambda)] - \Pr[1 \leftarrow \mathbf{Hyb}_1^{\mathcal{A}}(1^\lambda)]| \leq \mathsf{negl}(\lambda)$ and $|\Pr[1 \leftarrow \mathbf{Hyb}_{4,QR}^{\mathcal{A}}(1^\lambda)] - \Pr[1 \leftarrow \mathbf{Hyb}_5^{\mathcal{A}}(1^\lambda)]| \leq \mathsf{negl}(\lambda)$.

Proof. In $\mathbf{Hyb}_0^{\mathcal{A}}(1^\lambda)$, k_t and Γ_t for each key query are sampled uniformly at random and independently of the set S and all other values. So, sampling them during the setup phase does not change the view of $\mathcal{A}$. By Lemma A.4, the probability that $|\{t \in [Q] : k_t = k\}| > \lambda$ for some $k \in [Q]$ is

negligible. Otherwise, for each $k \in [Q]$, the sets Γ_t with $k_t = k$ are at most λ independent and uniformly random subsets of $[R]$ of size $\lambda+1$. So, by Lemma A.3 and a union bound over $k \in [Q]$, the probability that $|B_k| \geq \lambda+1$ for some $k \in [Q]$ is at most $Q \cdot \mathsf{negl}(\lambda)$. As $Q = \mathsf{poly}(\lambda)$, the probability that the challenger aborts in $\mathbf{Hyb}_1^{\mathcal{A}}(1^\lambda)$ is negligible. Conditioned on the challenger not aborting, both hybrids are identical. The same argument applies to $\mathbf{Hyb}_{4,QR}^{\mathcal{A}}(1^\lambda)$ and $\mathbf{Hyb}_5^{\mathcal{A}}(1^\lambda)$. Thus, the lemma holds. □

Lemma A.8. If PPS satisfies payload hiding property (Definition 4.1), then for any $v \in [QR]$ and admissible PPT adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for any $\lambda \in \mathbb{N}$, $|\Pr[1 \leftarrow \mathbf{Hyb}_{2,v-1}^{\mathcal{A}}(1^\lambda)] - \Pr[1 \leftarrow \mathbf{Hyb}_{2,v}^{\mathcal{A}}(1^\lambda)]| \leq \mathsf{negl}(\lambda)$ and $|\Pr[1 \leftarrow \mathbf{Hyb}_{4,v-1}^{\mathcal{A}}(1^\lambda)] - \Pr[1 \leftarrow \mathbf{Hyb}_{4,v}^{\mathcal{A}}(1^\lambda)]| \leq \mathsf{negl}(\lambda)$.

Proof. Let (k^*, r^*) be the v-th pair of $[Q]\times[R]$ in lexicographic order. If $r^* \in B_{k^*}$, then $\mathbf{Hyb}_{2,v-1}^{\mathcal{A}}(1^\lambda)$ and $\mathbf{Hyb}_{2,v}^{\mathcal{A}}(1^\lambda)$ are identical. Assume for the sake of contradiction that there exists an adversary $\mathcal{A}$ such that $|\Pr[1 \leftarrow \mathbf{Hyb}_{2,v-1}^{\mathcal{A}}(1^\lambda)] - \Pr[1 \leftarrow \mathbf{Hyb}_{2,v}^{\mathcal{A}}(1^\lambda)]| = \epsilon(\lambda)$ for a non-negligible function $\epsilon(\cdot)$. We construct a reduction adversary $\mathcal{B}$ that breaks the payload hiding property of PPS with non-negligible advantage. Description of $\mathcal{B}$ is as follows.

$\mathcal{B}$ receives $(N, 1^Q, i^*, \mu_0, \mu_1)$ from $\mathcal{A}$. It samples (k_t, Γ_t) for each $t \in [Q]$, computes B_k for each $k \in [Q]$, and aborts as in $\mathbf{Hyb}_1^{\mathcal{A}}(1^\lambda)$. If $r^* \in B_{k^*}$, $\mathcal{B}$ simulates $\mathbf{Hyb}_{2,v-1}^{\mathcal{A}}(1^\lambda)$ on its own and outputs its output. Otherwise, $\mathcal{B}$ samples the shares $(\sigma_{k,r}^*)_{k,r}$ of μ_0 as in $\mathbf{Hyb}_0^{\mathcal{A}}(1^\lambda)$ and sends $(N, i^*, \sigma_{k^*,r^*}^*)$ to the PPS payload hiding challenger. It receives $\mathsf{pps.crs}$ and $\mathsf{pps.ct}^\circ$, and sets $\mathsf{pps.crs}_{k^*,r^*} := \mathsf{pps.crs}$ and $\mathsf{pps.ct}_{k^*,r^*}^* := \mathsf{pps.ct}^\circ$. For each $(k,r) \neq (k^*, r^*)$, $\mathcal{B}$ samples $\mathsf{pps.crs}_{k,r}$, $\mathsf{pps.msk}_{k,r}$, and $\mathsf{pps.ct}_{k,r}^*$ as in $\mathbf{Hyb}_{2,v-1}^{\mathcal{A}}(1^\lambda)$. $\mathcal{B}$ sends $(\mathsf{pp}, \mathsf{ct}^*)$ to $\mathcal{A}$ and answers the queries of $\mathcal{A}$ as follows.

- Given (Enc, i, μ) with $i \neq i^*$, $\mathcal{B}$ samples the shares $(\sigma_{k,r})_{k,r}$ of μ as in $\mathbf{Hyb}_0^{\mathcal{A}}(1^\lambda)$. It queries the PPS challenger with (i, σ_{k^*,r^*}) to receive $\mathsf{pps.ct}_{k^*,r^*}$, and samples $\mathsf{pps.ct}_{k,r} \leftarrow \mathsf{PPS.skEnc}(\mathsf{pps.msk}_{k,r}, i, \sigma_{k,r})$ for each $(k,r) \neq (k^*, r^*)$. It sends $\mathsf{ct} := (\mathsf{pps.ct}_{k,r})_{k,r}$ to $\mathcal{A}$.
- Given the t-th query (KGen, S) with $i^* \notin S$, for each $r \in \Gamma_t$, if $(k_t, r) = (k^*, r^*)$, $\mathcal{B}$ sends S to the PPS challenger and receives $\mathsf{pps.sk}_{S,r}$. Otherwise, it samples $\mathsf{pps.sk}_{S,r} \leftarrow \mathsf{PPS.KGen}(\mathsf{pps.msk}_{k_t,r}, S)$. $\mathcal{B}$ sends $\mathsf{sk}_S := (k_t, \Gamma_t, (\mathsf{pps.crs}_{k_t,r}, \mathsf{pps.sk}_{S,r})_{r\in\Gamma_t})$ to $\mathcal{A}$.

At the end, if $\mathcal{B}$ has not sent any set to the PPS challenger, it sends $\emptyset$ and ignores the response. $\mathcal{B}$ outputs the guess b' of $\mathcal{A}$ as its own.

As $r^* \notin B_{k^*}$, at most one key query uses $\mathsf{pps.msk}_{k^*,r^*}$. So, $\mathcal{B}$ sends at most one set to the PPS challenger and this set does not contain i^*. Moreover, every ciphertext query of $\mathcal{B}$ is on an index $i \neq i^*$, and the PPS challenger answers ciphertext queries made after the set is sent by Remark 6.15. So, $\mathcal{B}$ is a valid payload hiding adversary against PPS. If the PPS challenger samples $\mathsf{pps.ct}^\circ \leftarrow \mathsf{PPS.skEnc}(\mathsf{pps.msk}, i^*, \sigma_{k^*,r^*}^*)$, then $\mathcal{B}$ behaves like $\mathbf{Hyb}_{2,v-1}^{\mathcal{A}}(1^\lambda)$ with $\mathsf{pps.msk}_{k^*,r^*} := \mathsf{pps.msk}$. Otherwise, $\mathsf{pps.ct}^\circ \leftarrow \mathsf{PPS.pkEnc}(i^*)$ and $\mathcal{B}$ behaves like $\mathbf{Hyb}_{2,v}^{\mathcal{A}}(1^\lambda)$. Thus, $\mathcal{B}$ breaks the payload hiding property of PPS with advantage $\epsilon(\lambda)$. The argument for $\mathbf{Hyb}_{4,v-1}^{\mathcal{A}}(1^\lambda)$ and $\mathbf{Hyb}_{4,v}^{\mathcal{A}}(1^\lambda)$ is the same in the opposite direction with the shares of μ_1 in place of the shares of μ_0. Thus, the lemma holds. □

Lemma A.9. $\Pr[1 \leftarrow \mathbf{Hyb}_{2,QR}^{\mathcal{A}}(1^\lambda)] = \Pr[1 \leftarrow \mathbf{Hyb}_3^{\mathcal{A}}(1^\lambda)]$.

Proof. In $\mathbf{Hyb}^{\mathcal{A}}_{2,QR}(1^\lambda)$, for each $k \in [Q]$, the shares $\sigma^*_{k,r}$ for $r \notin B_k$ are not used anywhere. So, the shares of μ_0 are only used through $(\sigma^*_{k,r})_{r\in B_k}$ for each $k \in [Q]$. As the challenger does not abort, $|B_k| \le \lambda$. So, by privacy of Shamir secret sharing (Lemma A.2), $(\sigma^*_{k,r})_{r\in B_k}$ is uniformly random and independent of the shared value for each $k \in [Q]$, and these are independent across $k \in [Q]$ as each bucket uses independent randomness for $\mathsf{Shamir.Share}$. So, the distribution of all the values used in the experiment is the same whether μ_0 or μ_1 is shared. Thus, the lemma holds. □

Using these lemma, we prove the theorem. □

Corollary A.10. Assuming the hardness of LWE with sub-exponential modulus-to-noise ratio, for any $N \le 2^\lambda$ and polynomial $Q = Q(\lambda)$, there exists a Q-bounded robust PPS scheme (Definition A.1) for $\mathcal{M}_\lambda = \mathbb{F}$ with $|\mathsf{sk}_S| = \mathsf{poly}(\lambda, \log N)$ and $|\mathsf{ct}|, |\mathsf{pp}| = Q \cdot \mathsf{poly}(\lambda, \log N)$. Moreover, Decode reads only $\mathsf{poly}(\lambda, \log N)$ bits of ct and does not read pp.

Proof. By Corollary 4.30, there exists a PPS scheme with full compactness for message space $\{0,1\}^\lambda$ assuming LWE with sub-exponential modulus-to-noise ratio, and it satisfies payload hiding against post-key ciphertext queries by Remark 6.15. As $h = O(\log \lambda) \le \lambda$, we can encrypt elements of $\mathbb{F}$ by padding them to λ bits. Plugging this into Construction A.5 and using Theorem A.6 along with the correctness, compactness, and locality of Construction A.5, we get the corollary. □

A.2 Secret Sharing for Evolving Graphs from PPS

Plugging Construction A.5 into Construction 6.17 with $\mathsf{RPPS.Setup}(1^\lambda, 2^\lambda, 1^Q)$ gives a CSS scheme for evolving graphs with collusion bound Q, as discussed in the bounded variants of Section 6.2.1. By locality, Recon never reads $\mathsf{sh}_0 = \mathsf{pp}$, as every secret key carries the common random strings it needs. So, the dealer outputs an empty public share instead. The share of a vertex consists of a secret key of size $\mathsf{poly}(\lambda)$ and a ciphertext of size $Q \cdot \mathsf{poly}(\lambda)$.

Theorem A.11. If PPS satisfies payload hiding (Definition 4.1) against post-key ciphertext queries (Remark 6.15), then Construction 6.17 instantiated with Construction A.5 is an adaptively secure CSS scheme for evolving graphs (Definition 6.16) with collusion bound Q and unbounded degree where the shares have size $Q \cdot \mathsf{poly}(\lambda)$ and there is no public share.

Proof. This follows from Theorem A.6, Theorem 6.19, and the bounded variants of Section 6.2.1, where the proof of Lemma 6.20 is adapted to bounded adaptive robustness as described there. Removing sh_0 does not affect correctness by locality and only reduces the view of $\mathcal{A}$. □

Graphs Fixed in Advance. For a graph G with vertex set $[L]$ that is fixed in advance, we run $\mathsf{RPPS.Setup}(1^\lambda, L, 1^Q)$ instead, so that the index space is $[L]$ and the shares have size $Q \cdot \mathsf{poly}(\lambda, \log L)$, and add the vertices in order. As the corrupted vertices form an independent set of G, it suffices to set Q to an upper bound on the independence number of G.

Corollary A.12. Assuming a PPS scheme (Definition 4.1, Remark 6.15), for any graph G with L vertices and independence number at most Q, there exists an adaptively secure CSS scheme for the graph access structure of G with no public share and share size $Q \cdot \mathsf{poly}(\lambda, \log L)$.

For bipartite graphs, we can do better by giving secret keys to one side and ciphertexts to the other.

Corollary A.13. Assuming a PPS scheme (Definition 4.1, Remark 6.15), for any bipartite graph $G = (\mathsf{L} \cup \mathsf{R}, E)$ with L vertices and $d := \max(1, \max_{v \in \mathsf{L}} |\mathsf{R} \setminus N(v)|)$, there exists an adaptively secure CSS scheme for the graph access structure of G with no public share where each vertex in L has share size $d \cdot \mathsf{poly}(\lambda, \log L)$ and each vertex in R has share size $\mathsf{poly}(\lambda, \log L)$.

Proof. We run $\mathsf{RPPS.Setup}(1^\lambda, L, 1^d)$, give each $u \in \mathsf{L}$ the share $\mathsf{ct}_u \leftarrow \mathsf{skEnc}(\mathsf{msk}, u, s)$, and give each $v \in \mathsf{R}$ the share $\mathsf{sk}_v \leftarrow \mathsf{KGen}(\mathsf{msk}, N(v))$. Along an edge $\{u, v\}$ with $u \in \mathsf{L}$ and $v \in \mathsf{R}$, $\mathsf{Decode}(N(v), \mathsf{sk}_v, u, \mathsf{ct}_u)$ recovers s. The share sizes follow from Corollary A.10 and locality. For security, we follow the proof of Theorem 6.19 with hybrids over $j \in \mathsf{L}$ only, as only the vertices in L hold ciphertexts. If $\mathcal{A}$ corrupts no vertex in L, its view consists of secret keys only and is independent of s. Otherwise, in the reduction for $j \in \mathsf{L}$, every corrupted $v \in \mathsf{R}$ satisfies $j \notin N(v)$, so $\mathcal{B}$ makes at most $|\mathsf{R} \setminus N(j)| \leq d$ queries of the form $(\mathsf{KGen}, \cdot)$, and bounded adaptive robustness with $Q := d$ suffices. □

Both corollaries improve the share size $\sqrt{L} \cdot \mathsf{poly}(\lambda)$ of [ABI+23, Corollary 4.2] from LWE for the respective graph classes, and, unlike the PPRGs from LWE in [ABI+23, HWW25], they require no public share and extend to evolving graphs by Theorem A.11.

Acknowledgments

Saikumar Yadugiri would like to thank Jiaqi Cheng for initial discussions on succinct functional encryption.

Statement on AI Usage. All the technical ideas and proofs in the paper were crafted by humans. AI tools including Claude Opus 5.5 and GPT 6.1 Sol were used to create an initial draft of the paper. The human authors then verified the correctness of all the proofs. The human authors take full responsibility for correctness of all claims and results in this work.