Deletions and Dishonesty: Probabilistic Data Structures in Adversarial Settings

Mia Filić[1], Keran Kocher[1] , Ella Kummer[1], and Anupama Unnikrishnan[1]

[1]ETH Zurich
`mfilic@ethz.ch`

Abstract. Probabilistic data structures (PDS) are compact representations of high-volume data that provide approximate answers to queries about the data. They are commonplace in today's computing systems, finding use in databases, networking and more. While PDS are designed to perform well under benign inputs, they are frequently used in applications where inputs may be adversarially chosen. This may lead to a violation of their expected behaviour, for example an increase in false positive rate.

In this work, we focus on PDS that handle approximate membership queries (AMQ). We consider adversarial users with the capability of making adaptive insertions, deletions and membership queries to AMQ-PDS, and analyse the performance of AMQ-PDS under such adversarial inputs.

We argue that deletions significantly empower adversaries, presenting a challenge to enforcing honest behaviour when compared to insertion-only AMQ-PDS. To address this, we introduce a new concept of an honest setting for AMQ-PDS with deletions. By leveraging simulation-based security definitions, we then quantify how much harm can be caused by adversarial users to the functionality of AMQ-PDS. Our resulting bounds only require calculating the maximal false positive probability and insertion failure probability achievable in our novel honest setting.

We apply our results to Cuckoo filters and Counting filters. We show how to protect these AMQ-PDS at low cost, by replacing or composing the hash functions with keyed pseudorandom functions in their construction. This strategy involves establishing practical bounds for the probabilities mentioned above. Using our new techniques, we demonstrate that achieving security against adversarial users making both insertions *and* deletions remains practical.

Keywords: probabilistic data structures · Counting filters · Cuckoo filters · security · simulation-based proofs

1 Introduction

Probabilistic data structures (PDS) are widespread in today's data-driven world. They find a multitude of uses across our computing systems, in databases, networking and communication. By compactly representing data, they offer improved efficiency, with the tradeoff of providing approximate (rather than exact) answers to queries about the data.

Each PDS is specifically designed to answer certain kinds of queries. An important category of PDS is those providing approximate answers to membership queries, i.e. "is an element x a member of a set S?". We refer to this category as AMQ-PDS, which are the focus of this work. Examples of AMQ-PDS include Bloom filters [5], Counting filters [12] and Cuckoo filters [10]. While Bloom filters only support insertions of elements into the set, Counting and Cuckoo filters also allow elements to be deleted. Other categories of PDS include frequency estimators such as Count-Min sketches [9] and Heavy Keepers [17], and cardinality estimators such as HyperLogLog [14] and KMV sketches [3].

PDS find a myriad of applications, from estimating the number of distinct Google search queries [19] and detecting anomalies in network traffic [17, 22] to building privacy-preserving recommendation systems [28]. AMQ-PDS are beneficial for database query speedup [34], spam detection [38], resource and packet routing in networks [6], certificate revocation systems [23], DNA sequence analysis [37, 29], and more [26]. In particular, AMQ-PDS that support deletions, such as Counting and Cuckoo filters, are useful for cache sharing among web proxies [12], speedup of post-quantum TLS handshakes [36], efficient certificate revocation checking [35], mobile private contact discovery [20, 18], and fighting fake news [25].

The wide deployment of PDS across applications, however, comes with an increasing risk of adversarial interference. By carefully choosing inputs, malicious users can force specific elements to become false positives in AMQ-PDS [16], cause frequencies of elements to be overestimated [8, 27], or artificially inflate cardinality estimates [33], for example. This leads to dangerous consequences for the use of PDS in practice. In spite of this, such adversarial settings are typically not covered by the performance guarantees of PDS; their expected behaviour is characterised assuming honest inputs. To protect PDS against adversarial influence, cryptographic techniques can be a powerful tool. Combining PDS with cryptography results in a significant new research area with many critical open questions.

1.1 Our Contributions

In this paper, we study the correctness of AMQ-PDS in adversarial settings. We focus on malicious users interacting with an AMQ-PDS hosted by an honest service provider. In practice, users interact with the AMQ-PDS through an API, allowing dynamic updates to the stored dataset, through insertions and deletions, as well as membership queries. In this work, we address how malicious users can leverage adaptive insertions, membership queries *and* deletions to manipulate the performance of AMQ-PDS. We will argue that deletions, in particular, are a powerful tool for adversaries. While we focus on two commonly used AMQ-PDS, Cuckoo filters [10] and Counting filters [12], our definitions are general and can be applied to a broad range of AMQ-PDS.

Syntax for AMQ-PDS. Inspired by [13, 8], we establish a syntax for AMQ-PDS that support insertions, deletions and membership queries. We identify consistency rules for the behaviour of AMQ-PDS, satisfied by Counting and Cuckoo filters, that will allow us to prove results on their adversarial correctness.

Simulation-based framework. We employ a simulation-based approach [24] to define security, following recent work [13, 33]. In this approach, the adversary is modelled as interacting with the AMQ-PDS in either a "real world" or an "ideal world". In the real world, the adversary has access to the AMQ-PDS through an API that allows it to insert and delete items, and make membership queries. In the ideal world, the adversary instead interacts with a simulator that models honest behaviour of the AMQ-PDS. At the end of its execution, the adversary produces an output, which is used to distinguish between the two worlds. By quantifying the distance between the worlds, we bound how much harm the adversary can do in the real world by relating it to the honest operation of the AMQ-PDS in the ideal world.

Simulation-based security definitions are traditionally used to analyse notions of privacy (for example, in searchable encryption [7]), where the simulator is given some leakage. By proving that the two worlds are indistinguishable, one concludes that the adversary can only learn this leakage, which is deemed acceptable. In contrast, our approach does not require indistinguishability between the worlds in order to give useful bounds; we will show how they can be used to set parameters for secure PDS in practice.

The power of the simulation-based approach in analysing correctness is that it covers all adversarial goals, in contrast to the game-based approach with a specific adversarial goal [8]. In practice, this means that one only needs to compute the probability of achieving a particular goal in the honest setting (which is well-studied in the PDS literature), in order to upper bound the probability of achieving it in the adversarial setting.

Adversarial correctness for AMQ-PDS with insertions and deletions. To analyse adversarial correctness using the simulation-based approach, the first question to address is how to define "honest" behaviour. Allowing deletions (in addition to insertions and membership queries), however, introduces substantial hurdles.

In [13], the notion of a non-adversarially-influenced (NAI) state was proposed for insertion-only AMQ-PDS. Intuitively, this captures the idea that the state of an AMQ-PDS can be thought of as honest if one cannot predict the effect of each insertion on the state, prior to the insertion. To achieve this for many prominent AMQ-PDS, one can replace the hash functions used in their constructions with keyed Pseudo-Random Functions (PRFs).

With deletions, however, the above idea no longer suffices to capture honesty. The ability to delete elements after inserting them means that an adversary could effectively reset the state if not satisfied. For example, consider cache summarisation for content routing [1, 12]. Here, an element is automatically added to or removed from the filter whenever the cache is updated. The cache's size poses a natural bound on the number of elements the filter stores. So, the attacker might want to force removal of elements that do not contribute to its goal of, for example, increasing the false positive probability (FPP) or making a specific target a false positive. The former significantly increases time for content retrieval on average, while the latter substantially increases retrieval time of the targeted content. While such a final state satisfy insertion unpredictability, it would still be adversarially influenced. Therefore, the deletion functionality of AMQ-PDS forms an intrinsic barrier to enforcing honesty.

Further, another complication arises from false negatives. While insertion-only AMQ-PDS may have false positives (elements that appear to be in the set when they have not been inserted), deletions may also lead to false negatives (elements that appear to not be in the set when they have not been deleted). The FPP of AMQ-PDS is typically well-characterised; false negatives, which can arise (for example) through deleting elements that were never inserted, are often assumed not to occur under honest operation. In an adversarial setting, we can no longer assume this.

We circumvent these obstacles by proposing a new notion of honesty for AMQ-PDS with both insertions and deletions, which we call NAI*. We show that building a simulator that satisfies NAI* suffices to analyse adversarial correctness for our AMQ-PDS of interest.

Our results show how to provably protect AMQ-PDS by replacing or composing public hash functions with PRFs and giving concrete bounds on the probability of achieving any adversarial goal through adaptive queries. Practitioners can use our concrete bounds to set AMQ-PDS parameters that guarantee security even with adversarial users. This is in contrast to how parameters are currently set in practice, with bounds on (for example) FPP being easily violated through precomputation attacks (on public hash functions). Using our results, practitioners can guarantee that FPP will stay below a certain threshold even with adaptive queries. This extends to any adversarial goal, e.g. creating false negatives, causing insertion failures. By showing how to ensure AMQ-PDS behave as expected even with malicious users, our work impacts any application of AMQ-PDS - in particular, applications requiring dynamic deletions, insertions and membership queries (e.g. cache sharing, coupon validation, etc.).

We emphasise that our focus is on users exploiting the API that allows interaction with an AMQ-PDS hosted by an honest service provider. To our knowledge, such an API typically does not allow users to view its internal state, e.g. [2]. Of course, in a different adversarial scenario with a compromised service provider, users could gain access to the state. While out of scope in this work, we later discuss why our results are not directly applicable to such a setting in Remark 3.

Analysis of Counting and Cuckoo filters. We conclude by providing a concrete evaluation of our security theorems by analysing Counting and Cuckoo filters. The usage of public hash functions in their original formulations leads to vulnerabilities from precomputation attacks [16, 8]. Using our theorems, we demonstrate how to provably protect them by replacing or composing the hash functions with PRFs (at the cost of needing secure key management). This requires deriving novel bounds on their NAI* false positive probability, as well as their NAI* insertion failure probability, both of which we show how to upper bound using results from the (insertion-only) AMQ-PDS literature.

Finally, we investigate the impact of our analysis for choosing appropriate parameters to secure AMQ-PDS in practice. Our results illustrate that protecting AMQ-PDS against adversarial users who can harness their full functionality is practical. Further, as a result of our new insights and techniques, extending the user's capabilities to include deletions does not compromise security.

1.2 Related Work

In [13], Filić *et al.* proposed a simulation-based framework for analysing the adversarial correctness and privacy of AMQ-PDS that only support insertions. By building a simulator that models the non-adversarial operation of AMQ-PDS, they derived bounds on the closeness of an adversarially generated state to that of an honest one, applying their framework to derive correctness guarantees for Bloom and insertion-only Cuckoo filters under adversarial inputs. In our work, we use a similar methodology but cover the full functionality of AMQ-PDS, i.e. allowing deletions as well as insertions. Thus, we solve an important question left unanswered by their work, resulting in a more complete analysis of adversarial correctness of AMQ-PDS.

A simulation-based approach was also employed in [33] to study the HyperLogLog cardinality estimator in adversarial settings. While our proof technique is conceptually similar, the types of queries supported by AMQ-PDS lead to more powerful adversarial strategies, and thus a more complicated analysis.

The work of Clayton *et al.* [8] focused on the adversarial correctness of AMQ-PDS Bloom and Counting filters. They examined an "l-thresholded" variant of Counting filters, where insertions are disallowed if more than ℓ counters are set. Their approach utilised a game-based formalism, which required defining a specific winning condition for the adversary, i.e. finding a certain number of false positives or false negatives. We provide a more detailed comparison of our work with [8] in Supplementary material C. A similar approach was adopted in [4] with an adversary who tries to maximise the false positive rate of AMQ-PDS by repeating membership queries. In contrast to these game-based methods, the simulation-based formalism does not require specifying an adversarial goal. This allows one to use our results to re-derive bounds for any specific adversary.

In [31, 32], Naor and Yogev studied the adversarial correctness of Bloom filters, again using a game-based approach. Recent work by Naor and Oved [30] further extended this to propose various robustness notions for Bloom filters. However, their adversarial model is more restricted than ours, without the ability to make adaptive insertions and membership queries. Further, as their focus is on Bloom filters, deletions do not play a role.

In [39], Yeo analysed Cuckoo hash tables, which are closely related to Cuckoo filters. However, they considered a static adversarial setting, where a set of elements is inserted at the start, with a specific adversarial goal of causing the insertion of this set to fail. In this work, we are interested in a more powerful setting where adversaries can dynamically update the dataset and can have any goal. Adversarial influence on the false positive rate of Cuckoo filters was studied in [21], but in a similarly restricted adversarial model.

Therefore, in comparison to previous work, we are the first to rigorously analyse adversarial correctness of Counting and Cuckoo filters in their full capability, for any adversarial goal. This fills a significant gap in the literature.

For scenarios where the data itself is sensitive, studying privacy might also become important. Leveraging the power of deletions to deduce information about elements in Counting filters, [15] proposed attacks on their privacy. This highlights an intrinsic challenge in enforcing privacy for AMQ-PDS with deletions, leaving the task an interesting open question.

1.3 Paper Organisation

We start with preliminaries in Section 2. In Section 3.1, we define the syntax for AMQ-PDS with deletions, the notion of a non-adversarial setting, and properties of our AMQ-PDS of interest. We analyse adversarial correctness in Section 4, and discuss the usefulness of our results in practice in Section 5.

2 Preliminaries

Notation. We follow the notation of [13], repeated here for clarity. For an integer $m \in \mathbb{Z}_{\geq 1}$, we write $[m]$ to denote the set $\{1, 2, ..., m\}$. We consider all logarithms to be in base 2. Given two sets $\mathfrak{D}$ and $\mathfrak{R}$, we define $\mathsf{Funcs}[\mathfrak{D}, \mathfrak{R}]$ to be the set of functions from $\mathfrak{D}$ to $\mathfrak{R}$. We write $F \leftarrow\!\!\$\ \mathsf{Funcs}[\mathfrak{D}, \mathfrak{R}]$ to mean that F is a random function $\mathfrak{D} \xrightarrow{F} \mathfrak{R}$. Given a set S, we denote the identity function over S as $\mathrm{Id}_S \colon S \to S$. For a probability distribution D, we write $x \leftarrow\!\!\$\ D$ to mean that x is sampled according to D. We define the statistical distance between two random variables X, Y with finite support $D = \mathrm{Supp}(X) = \mathrm{Supp}(Y)$ as $SD(X, Y) := \frac{1}{2} \sum_{z \in D} |\Pr[X = z] - \Pr[Y = z]|$. For a set S (resp. a list L), we denote by $|S|$ (resp. $|L|$) the number of elements in S (resp. L). A fixed-length list of length s initialised empty is denoted by $a \leftarrow \perp^s$. We denote by $\mathrm{load}(a)$ the number of set entries of a. To insert an entry x into the first unused slot in a we write $a' \leftarrow a \diamond x$ such that $a' = x \perp ... \perp$ with $s-1$ trailing $\perp$s and $\mathrm{load}(a') = 1$. A further insertion $a'' \leftarrow a' \diamond y$ results in $a'' = x\, y \perp ... \perp$ with $\mathrm{load}(a'') = 2$, and so on. We refer to the i-th entry in a list a as $a[i]$. In algorithms, we assume that all key-value stores are initialised with value $\perp$ at every index, using the convention that $\perp < n$, $\forall n \in \mathbb{R}$, and we denote it as $\{\}$. For a key-value store a, we refer to the value of the entry with key k as $a[k]$. We write variable assignments using $\leftarrow$, unless the value is output by a randomised algorithm, for which we use $\leftarrow\!\!\$$.

For a randomised algorithm alg, we write *output* $\leftarrow \mathsf{alg}(\mathrm{input}_1, \mathrm{input}_2, ..., \mathrm{input}_\ell; r)$, where $r \in \mathcal{R}$ denotes the coins that can be used by alg and $\mathcal{R}$ is the set of possible coins. We may also suppress coins whenever it is notationally convenient to do so. For a deterministic algorithm, r can be set to $\perp$. We remark that the output of a randomised algorithm can be seen as a random variable over

```
$Exp_R^{PRF}(\mathcal{B})$
1 $K \leftarrow_\$ \mathcal{K};\ F \leftarrow_\$ \mathrm{Func}[\mathfrak{D}, \mathfrak{R}]$
2 $b \leftarrow_\$ \{0,1\};\ b' \leftarrow_\$ \mathcal{B}^{\mathbf{RoR}}$
3 return $b'$

Oracle $\mathbf{RoR}(x)$
1 if $b = 0:\ y \leftarrow R_K(x)$
2 else $:\ y \leftarrow F(x)$
3 return $y$
```

Fig. 1: The PRF experiment.

the output space of the algorithm. Unless otherwise specified, we will consider random coins to be sampled uniformly from $\mathcal{R}$, independently from all other inputs and/or state, and refer to such r as "freshly sampled". If alg is given oracle access to functions f_1, ..., f_n, we denote it by $\mathsf{alg}^{f_1,\cdots f_n}$.

We will consider AMQ-PDS that can store elements from finite domains $\mathfrak{D}$ by letting $\mathfrak{D} = \cup_{\ell=0}^{L}\{0,1\}^{\ell}$ for some large but finite value of L, say $L = 2^{64}$. In our constructions, we will make use of pseudorandom functions, which we will model as truly random functions to which the AMQ-PDS has oracle access.

Definition 1. *Consider the PRF experiment in Fig. 1. We say a pseudorandom function family $R\colon \mathcal{K} \times \mathfrak{D} \to \mathfrak{R}$ is (q, t, ε)-secure if for all adversaries $\mathcal{B}$ running in time at most t and making at most q queries to its **RoR** oracle in Exp_R^{PRF},*

$$Adv_R^{PRF}(\mathcal{B}) := |\Pr[\, b' = 1 | b = 0\,] - \Pr[\, b' = 1 | b = 1\,]| \le \varepsilon.$$

We say $\mathcal{B}$ is a (q, t)-PRF adversary.

3 AMQ-PDS

In this section, we formalise the syntax of AMQ-PDS and their behaviour under non-adversarial inputs. We formally define our AMQ-PDS of interest, Counting and Cuckoo filters, and discuss some common properties that they satisfy.

3.1 Syntax

We now define the syntax of an AMQ-PDS, extending that of [13] to include deletions. Let Π be an AMQ-PDS. We denote its public parameters by pp, and its state as $\sigma \in \Sigma$, where Σ denotes the space of possible states of Π. The set of elements that can be inserted into Π is denoted by $\mathfrak{D}$, unless stated otherwise. We consider a syntax consisting of four algorithms:

- The setup algorithm $\sigma \leftarrow \mathsf{setup}(pp; r)$ sets up the initial state of an empty PDS with public parameters pp; it will always be called first to initialise the AMQ-PDS.
- The insertion algorithm $(b, \sigma') \leftarrow \mathsf{ins}(x, \sigma; r)$, given an element $x \in \mathfrak{D}$, attempts to insert it into the AMQ-PDS, and returns a bit $b \in \{\bot, \top\}$ representing whether the insertion was successful ($b = \top$) or not ($b = \bot$), and the state σ' of the AMQ-PDS after the insertion.

– The deletion algorithm $(b, \sigma') \leftarrow \mathsf{del}(x, \sigma)$, given an element $x \in \mathfrak{D}$, attempts to delete x from the AMQ-PDS, i.e. attempts to remove everything that a successful insertion on x added to σ. The algorithm return a bit $b \in \{\bot, \top\}$ representing whether the deletion was successful ($b = \top$) or not ($b = \bot$), and the state σ' of the AMQ-PDS after the deletion.
– The membership querying algorithm $b \leftarrow \mathsf{qry}(x, \sigma)$, given an element $x \in \mathfrak{D}$, returns a bit $b \in \{\bot, \top\}$ (approximately) answering whether x was previously inserted ($b = \top$) or not ($b = \bot$) into the AMQ-PDS.

We remark that we only consider AMQ-PDS where membership queries do not change the state of the AMQ-PDS; thus, qry does not need to output a new σ' value. This includes popular AMQ-PDS such as Counting and Cuckoo filters.

Due to the approximate nature of AMQ-PDS, qry calls may return a false positive result with a certain probability. That is, we may have $\top \leftarrow \mathsf{qry}(x, \sigma)$ even though no call $\mathsf{ins}(x, \sigma'; r)$ was made post setup and prior to the membership query. We refer to the probability $\Pr[\top \leftarrow \mathsf{qry}(x, \sigma) \mid x \text{ was not inserted into } \Pi]$ as the *false positive probability* of an AMQ-PDS Π. In addition, since Counting and Cuckoo filters support deletions, qry calls may return a false negative result, where we may have $\bot \leftarrow \mathsf{qry}(x, \sigma)$ even though an $\mathsf{ins}(x, \sigma'; r)$ call was made beforehand. We refer to the probability $\Pr[\bot \leftarrow \mathsf{qry}(x, \sigma) \mid x \text{ was inserted into } \Pi]$ as the *false negative probability* of an AMQ-PDS Π.

Moreover, the insertion algorithm may fail to insert an element, for example if the AMQ-PDS has reached capacity. We denote the probability $\Pr[(\bot, \sigma) \leftarrow\$ \mathsf{ins}(x, \sigma)]$ as the *insertion failure probability.*

3.2 AMQ-PDS under non-adversarial inputs

We now define the expected behaviour of AMQ-PDS in a non-adversarial setting, since we will later quantify how much the state of an AMQ-PDS can deviate from this under adversarial inputs. As in [13], we will focus on AMQ-PDS that satisfy the following properties of *function-decomposability* and *reinsertion invariance.*

Definition 2 (Function-decomposability [13]). *Let Π be an AMQ-PDS and let $F \leftarrow\$ \mathsf{Funcs}[\mathfrak{D}, \mathfrak{R}]$ with $\mathfrak{R} \subset \mathfrak{D}$ be a random function to which Π has oracle access. We say Π is* F-decomposable *if*

$$\begin{aligned} \mathsf{ins}^F(x, \sigma; r) &= \mathsf{ins}^{Id_{\mathfrak{R}}}(F(x), \sigma; r) \quad \forall x \in \mathfrak{D}, \sigma \in \Sigma, r \in \mathcal{R}, \\ \mathsf{del}^F(x, \sigma) &= \mathsf{del}^{Id_{\mathfrak{R}}}(F(x), \sigma) \quad \forall x \in \mathfrak{D}, \sigma \in \Sigma, \\ \mathsf{qry}^F(x, \sigma) &= \mathsf{qry}^{Id_{\mathfrak{R}}}(F(x), \sigma) \quad \forall x \in \mathfrak{D}, \sigma \in \Sigma, \end{aligned}$$

where $\mathsf{ins}^{Id_{\mathfrak{R}}}$, $\mathsf{del}^{Id_{\mathfrak{R}}}$ and $\mathsf{qry}^{Id_{\mathfrak{R}}}$ cannot internally evaluate F due to not having oracle access to it and F being truly random. Function-decomposability also applies to AMQ-PDS with oracle access to multiple functions.

Definition 3 (Reinsertion invariance [13]). *Let Π be an AMQ-PDS. We say Π is* reinsertion invariant *if for all $x \in \mathfrak{D}, \sigma \in \Sigma$ such that $\top \leftarrow \mathsf{qry}(x, \sigma)$, we have $(\top, \sigma') \leftarrow \mathsf{ins}(x, \sigma; r) \implies \sigma = \sigma' \ \forall r \in \mathcal{R}$.*

Reinsertion invariance is a natural property to expect from AMQ-PDS since they are designed to represent sets and not multisets. Note that if reinsertion invariance does not apply, simply repeatedly inserting a single element could lead to blocking of further insertions.

If a reinsertion-invariant AMQ-PDS contains multiple copies of the same element, deleting one copy will result in all other copies being deleted. However, reinsertion invariance does not require the state of the AMQ-PDS to remain unchanged if elements are reinserted after being deleted.

For an *insertion-only* AMQ-PDS satisfying function-decomposability and reinsertion invariance, the notion of a non-adversarially influenced state was proposed in [13]. We give an alternative (but equivalent) definition below.

Definition 4 (n-NAI state). *Let Π be an AMQ-PDS with public parameters pp using $F = Id_{\mathfrak{R}}$ satisfying reinsertion invariance, and let $\sigma \leftarrow \mathsf{setup}(pp)$. Let n be a non-negative integer. Let $X_1, ..., X_n \leftarrow\$ \mathfrak{R}$. Let L be the list of operations on σ, where $L = [\mathsf{ins}^{Id_{\mathfrak{R}}}(X_1, \sigma), ..., \mathsf{ins}^{Id_{\mathfrak{R}}}(X_n, \sigma)]$. Then, σ is an n-*NAI state.

We then give an alternative (but equivalent) definition of the NAI false positive probability from [13].

Definition 5 (NAI false positive probability). *Let Π be an AMQ-PDS with public parameters pp, using a random function $F : \mathfrak{D} \to \mathfrak{R}$ satisfying F-decomposability and reinsertion invariance. Let n be a non-negative integer. Define the NAI false positive probability after n distinct insertions as*

$$P_{\Pi,pp}(FP \mid n) := \Pr\left[\begin{array}{c} \sigma \leftarrow \mathsf{setup}(pp) \\ \textit{for } i \in [n] : (b, \sigma) \leftarrow\$ \mathsf{ins}^{Id_{\mathfrak{R}}}(X_i \leftarrow\$ \mathfrak{R}, \sigma) : \\ \top \leftarrow\$ \mathsf{qry}^{Id_{\mathfrak{R}}}(X \leftarrow\$ \mathfrak{R}, \sigma) \end{array}\right].$$

Remark 1. Defs. 4 and 5 are equivalent to that of [13, Def. 3.4] for F-decomposable AMQ-PDS. Sampling n distinct elements from $\mathfrak{D}$ is equivalent to sampling n strings $X \leftarrow\$ \mathfrak{R}$. Similarly, sampling the queried element from $\mathfrak{D} \setminus V$, where V is the set of n inserted elements, is equivalent to sampling $X \leftarrow\$ \mathfrak{R}$.

As mentioned, the NAI state constructed in Def. 4 captures honesty for insertion-only AMQ-PDS. As long as the effect of every insertion on the state is unpredictable, the final state cannot deviate from "honest". However, for AMQ-PDS that also allow deletions, defining an honest setting is more involved. The deletion capability means that a user could insert elements, observe their effects, and then decide whether to delete them, i.e. to reset the state if not satisfied. In other words, even if every insertion is unpredictable, the final state may still be adversarially influenced (i.e. no longer an NAI state).

We overcome these issues with a new definition of the non-adversarial setting for function-decomposable, reinsertion-invariant AMQ-PDS, which we call *NAI**. NAI* captures honesty up to the extent that can be achieved with both insertions and deletions. We will show that the final state of the AMQ-PDS satisfying NAI* suffices to capture a non-adversarial setting that we can analyse using results from the PDS literature.

A key component of NAI* will be the following notion: for any element not previously inserted, the effect of its insertion on the state is unpredictable (*insertion unpredictability*). Intuitively, this can be thought of as replacing every insertion of an element $x \in \mathfrak{D}$ with $X \in \mathfrak{R}$ sampled uniformly at random. This is not necessarily ensured only by F-decomposability, since the interplay between $\mathsf{ins}, \mathsf{del}$ and qry on the same input could reveal information about F. We define insertion unpredictability in Def. 6.

Definition 6 (Insertion unpredictability). *Let Π be an AMQ-PDS with public parameters pp, using a random function $F : \mathfrak{D} \leftarrow \mathfrak{R}$, and satisfying F-decomposability and reinsertion invariance. Let $\sigma \leftarrow \mathsf{setup}(pp)$. Let $\{z_i\}$ be the elements that are successfully inserted into σ. For every first insertion of z_i, let $(\top, \sigma') \leftarrow \mathsf{ins}^F(z_i, \sigma_i)$ and $(\top, \overline{\sigma}) \leftarrow \mathsf{ins}^{Id_{\mathfrak{R}}}(X \leftarrow_{\$} \mathfrak{R}, \sigma_i)$. We say σ has* insertion unpredictability *if $SD\big(\sigma', \overline{\sigma}\big) = 0$.*

We are now ready to define an n-NAI* state. Although an NAI* state of an AMQ-PDS can be constructed through both insertions and deletions of elements, our definition will require that all insertions are unpredictable, deletions only happen on currently inserted elements, and repeated insertions of elements only change the state if that element has been deleted. These requirements essentially capture what we would expect from honest insertions and deletions on function-decomposable, reinsertion-invariant AMQ-PDS.

Definition 7 (n-NAI* state). *Let Π be an AMQ-PDS with public parameters pp using $F = Id_{\mathfrak{R}}$ satisfying reinsertion invariance, and let $\sigma \leftarrow \mathsf{setup}(pp)$. Let n be a non-negative integer. Let $X_1, ..., X_n \leftarrow_{\$} \mathfrak{R}$. Let L be the list of operations on σ, where each item in L is either $\mathsf{ins}^{Id_{\mathfrak{R}}}(\cdot, \sigma)$ or $\mathsf{del}^{Id_{\mathfrak{R}}}(\cdot, \sigma)$ on $X_1, ..., X_n$. Then, σ is an n-*NAI* state *if:*

- *for all X_i there is an operation in L equal to $\mathsf{ins}^{Id_{\mathfrak{R}}}(X_i, \sigma)$,*
- *for all successful $\mathsf{del}^{Id_{\mathfrak{R}}}(X_i, \sigma)$ operations in L, the preceding successful operation in L on X_i is $\mathsf{ins}^{Id_{\mathfrak{R}}}(X_i, \sigma)$,*
- *all successful $\mathsf{ins}^{Id_{\mathfrak{R}}}(X_i, \sigma)$ operations in L for which any prior successful operation in L on X_i is $\mathsf{ins}^{Id_{\mathfrak{R}}}(X_i, \sigma)$ either do not change the state, or have $\mathsf{del}^{Id_{\mathfrak{R}}}(X_i, \sigma)$ as their preceding successful operation on X_i in L.*

It is clear to see that every n-NAI state (Def. 4) is then an n-NAI* state. We now give an analogous formulation of Def. 7 for F-decomposable AMQ-PDS, where unsuccessful insertions do not change the state.

Corollary 1. *Let Π be an AMQ-PDS with public parameters pp using a random function $F : \mathfrak{D} \rightarrow \mathfrak{R}$ satisfying F-decomposability and reinsertion invariance, where unsuccessful insertions do not change the state. Let $\sigma \leftarrow \mathsf{setup}(pp)$ and n be a non-negative integer. Let L be the list of operations on σ, where each item in L is either $\mathsf{ins}^F(\cdot, \sigma)$ or $\mathsf{del}^F(\cdot, \sigma)$. Then, σ is an n-*NAI* state *if:*

– *it satisfies Def. 6,*
– *there are n distinct elements $\{z_i\}_{i\in[n]}$ for which an operation in L on z_i is $\mathsf{ins}^F(z_i,\sigma)$,*
– *for all successful $\mathsf{del}^F(z_i,\sigma)$ operations in L, the preceding successful operation in L on z_i is $\mathsf{ins}^F(z_i,\sigma)$, and*
– *all $\mathsf{ins}^F(z_i,\sigma)$ operations in L for which any prior successful operation in L on z_i is $\mathsf{ins}^F(z_i,\sigma)$ either do not change the state, or have $\mathsf{del}^F(z_i,\sigma)$ as their preceding successful operation on z_i in L.*

A natural next step would be to define the false positive probability and insertion failure probability for NAI* states, analogous to that of the insertion-only setting [13]. However, while deleting an inserted element may be an operation allowed under NAI*, a user could insert elements, observe their effects, and then decide whether to delete them, i.e. to reset the state. This means that, using only n distinct elements, a user can create many different NAI* states. Therefore, a more useful notion for NAI* states is the *maximal* false positive and insertion failure probability, defined in terms of the "worst possible" NAI* state.

Definition 8 (Maximal NAI* false positive probability). *Let Π be an AMQ-PDS with public parameters pp using a random function $F : \mathfrak{D} \to \mathfrak{R}$ satisfying F-decomposability and reinsertion invariance. Let n be a non-negative integer. Define the maximal NAI* false positive probability after n insertions as*

$$P^*_{\Pi,pp}(FP \mid n) := \Pr\left[\begin{array}{c} X_1,...,X_n \leftarrow\!\!\$\ \mathfrak{R} \\ \sigma \leftarrow \mathcal{U}^{Id_{\mathfrak{R}}}_{\Pi,pp}(X_1,...,X_n): \\ \top \leftarrow\!\!\$\ \mathsf{qry}^{Id_{\mathfrak{R}}}(X \leftarrow\!\!\$\ \mathfrak{R},\sigma) \end{array}\right],$$

where $\mathcal{U}^{Id_{\mathfrak{R}}}_{\Pi,pp}(X_1,...,X_n)$ outputs an NAI state created using $\mathsf{ins}^{Id_{\mathfrak{R}}}(\cdot,\sigma)$,$\mathsf{del}^{Id_{\mathfrak{R}}}(\cdot,\sigma)$ on $X_1,...,X_n$ that has the maximal false positive probability.*

Def. 8 captures the false positive probability of the "worst possible" NAI* state that can be created with insertions and deletions. The algorithm $\mathcal{U}$ gets n strings sampled uniformly at random from $\mathfrak{R}$ as input, and finds the ordering of insertions and deletions of these strings (possibly excluding some) that maximises the false positive probability. Since the queried $X \leftarrow\!\!\$\ \mathfrak{R}$ is sampled randomly, $\mathcal{U}$ is not increasing the probability that a particular element is a false positive; rather, it is creating a state with the highest false positive probability in general.

Definition 9 (Maximal NAI* insertion failure probability). *Let Π be an AMQ-PDS with public parameters pp, using a random function $F : \mathfrak{D} \to \mathfrak{R}$ satisfying F-decomposability and reinsertion invariance. Let n be a non-negative integer. Define the maximal NAI* insertion failure probability within n insertions as*

$$P^*_{\Pi,pp}(IF \mid n) := \Pr\left[\begin{array}{c} X_1,...,X_n \leftarrow\!\!\$\ \mathfrak{R} \\ \sigma \leftarrow \mathcal{V}^{Id_{\mathfrak{R}}}_{\Pi,pp}(X_1,...,X_n): \\ \textit{for some } l \in [n],\ (\bot,\sigma) \leftarrow\!\!\$\ \mathsf{ins}^{Id_{\mathfrak{R}}}(X_l,\sigma) \end{array}\right],$$

where $\mathcal{V}^{Id_{\mathfrak{R}}}_{\Pi,pp}(X_1,\ldots,X_n)$ *outputs an NAI* state created using* $\mathsf{ins}^{Id_{\mathfrak{R}}}(\cdot,\sigma)$, $\mathsf{del}^{Id_{\mathfrak{R}}}(\cdot,\sigma)$ *on* $X_1,\ldots,X_n$ *that has the maximal probability of an insertion on one of* $X_1,\ldots,X_n$ *failing.*

Def. 9 captures the insertion failure probability of the "worst possible" NAI* state that can be created with insertions and deletions. The algorithm $\mathcal{V}$ gets as an input n strings sampled uniformly at random from $\mathfrak{R}$, and then finds the ordering of insertions and deletions of these strings (possibly excluding some) that maximises the probability that inserting one of these strings will fail. Note that the definition is of a slightly different flavour to Def. 8; $\mathcal{V}$ can optimise its output in respect to X_l that is most likely to result in an insertion failure. Def. 9 naturally extends upon insertion failure definitions found in the literature [10, 12], where the probability is defined as one among n insertions failing.

3.3 Counting filters

Counting filters are an extension of the popular Bloom filters, with the added capability of supporting deletions of elements. A Counting filter consists of an array of counters σ of length m initially set to 0^m, and a family of k independent hash functions $H_i : \{0,1\}^* \to [m]$, for $i \in [k]$. To insert an element x into the filter, all k counters $H_i(x)$ of σ are incremented; if any counter reaches the maximum value $maxVal$, the insertion fails. To delete an element x from the filter, if all k counters $H_i(x)$ are greater than zero, they are all decremented; if not, the deletion fails. A membership query on x returns $\top$ if all k counters $H_i(x)$ are greater than zero. Due to collisions in the hash functions H_i, Counting filters can have both false positives and false negatives. As in [13], we will bundle the k hash functions H_i into a single function $F : \mathfrak{D} \to [m]^k$.

We now formally define Counting filters.

Definition 10. *Let* $m,k,maxVal$ *be positive integers. We define an* $(m,k,maxVal)$*-Counting filter to be the AMQ-PDS with algorithms defined in Fig. 2, with* $pp = (m,k,maxVal)$*, and* $F : \mathfrak{D} \to [m]^k$*.*

We recall from the literature a bound on the NAI false positive probability for Counting filters. Due to their membership query algorithm only checking for non-zero counters, as in the case of Bloom filters, this bound is the same for both Counting and Bloom filters.

Lemma 1. *([12],[13, Lemma 3.7]) Let* Π *be an* $(m,k,maxVal)$*-Counting filter using a random function* $F : \mathfrak{D} \to [m]^k$*. Then, for any* n*,* $P_{\Pi,pp}(FP \mid n) \leq \left[1 - e^{-\frac{(n+0.5)k}{m-1}}\right]^k$*.*

We now derive upper bounds on the maximal NAI* false positive probability and the maximal NAI* insertion failure probability for Counting filters.

Lemma 2. *Let* Π *be an* $(m,k,maxVal)$*-Counting filter using a random function* $F : \mathfrak{D} \to [m]^k$*. Then, for any* n*,* $P^*_{\Pi,pp}(FP \mid n) \leq \left[1 - e^{-\frac{(n+0.5)k}{m-1}}\right]^k$*.*

```
setup(pp)
1  $m,\ k,\ maxVal \leftarrow pp$
2  $\sigma \leftarrow 0^m$
3  return $\sigma$

qry$^F(x, \sigma)$
1  $(p_1, \ldots, p_k) \leftarrow F(x)$
2  for $i \in [k]$
3      if $\sigma[p_i] = 0$
4          return $\bot$
5  return $\top$

ins$^F(x, \sigma)$
1  $(p_1, \ldots, p_k) \leftarrow F(x)$
2  $a \leftarrow \top;\ c \leftarrow \{\}$
3  for $i \in [k]$
4      if $\sigma[p_i] = 0$
5          $a \leftarrow \bot$
6  if $a = \top$ : return $\top, \sigma$
7  for $i \in [k]$
8      if $c[p_i] = \bot$
9          $c[p_i] \leftarrow 0$
10     $c[p_i]$ += 1
11 for $p \in c$.keys()
12     // $p$ s.t. $c[p] \neq \bot$
13     if $\sigma[p]$+$c[p] > maxVal$
14         return $\bot, \sigma$
15 for $p \in c$.keys() :
16     $\sigma[p]$ += $c[p]$
17 return $\top, \sigma$

del$^F(x, \sigma)$
1  $(p_1, \ldots, p_k) \leftarrow F(x)$
2  $c \leftarrow \{\}$
3  for $i \in [k]$
4      if $c[p_i] = \bot$
5          $c[p_i] \leftarrow 0$
6      $c[p_i]$ += 1
7  for $p \in c$.keys()
8      // $p$ s.t. $c[p] \neq \bot$
9      if $\sigma[p] - c[p] < 0$
10         return $\bot, \sigma$
11 for $p \in c$.keys() :
12     $\sigma[p]$ −= $c[p]$
13 return $\top, \sigma$
```

Fig. 2: AMQ-PDS syntax instantiation for the Counting filter.

Proof (sketch). We construct an algorithm that inserts all $X_1, \ldots, X_n$ with *maxVal* set to ∞, and show that the false positive probability of the resulting state (which follows from Lemma 1) is an upper bound on the false positive probability of any n-NAI* state. For the full proof, see Supplementary Material.

Lemma 3. *Let Π be an $(m, k, maxVal)$-Counting filter using a random function $F : \mathfrak{D} \to [m]^k$. Then, for any n, $P^*_{\Pi,pp}(IF \mid n) \leq m \cdot \left[\frac{e \cdot n \cdot k}{maxVal \cdot m}\right]^{maxVal}$.*

Proof (sketch). We construct an algorithm that inserts all $X_1, \ldots, X_n$, using a modified insertion algorithm that always increments counters (i.e. the check in line 6 of the ins algorithm in Fig. 2 is skipped), and with *maxVal* set to ∞. Let the resulting state be denoted by Δ. We show that the insertion failure probability of any n-NAI* state with *maxVal* equal to some *limit* can be upper bounded by the probability that any counter in Δ exceeds *limit*. For the full proof, see Supplementary Material.

3.4 Cuckoo filters

Cuckoo filters were proposed as an alternative to Bloom filters with improved performance and support for deletions [10]. A Cuckoo filter consists of a collection

$(\sigma_i)_i$ of 2^{λ_I} buckets, each indexed by $i \in [2^{\lambda_I}]$ and containing s slots, together with a stash σ_{stash} containing one slot. They use two hash functions $H_I : \mathfrak{D} \rightarrow \{0,1\}^{\lambda_I}$ and $H_T : \mathfrak{D} \rightarrow \{0,1\}^{\lambda_T}$. To insert (resp. delete) an element x into the filter, its tag is computed as $H_T(x)$ and inserted (resp. deleted) into its first or second bucket, whose indices are computed as $i_1 = H_I(x), i_2 = i_1 \oplus H_I(H_T(x))$ respectively. If both buckets are full, an eviction process begins. A membership query on x returns $\top$ if $H_T(x)$ is found in either of its corresponding buckets or the stash. As for Counting filters, membership queries can return both false positive and false negative responses.

In [13], a variant of the standard Cuckoo filter called the *PRF-wrapped Cuckoo filter* was proposed, which was required for the proofs of adversarial correctness and privacy. In this variant, inputs to the $\mathsf{ins}, \mathsf{del}$ and qry algorithms are simply preprocessed with a random function $F : \mathfrak{D} \rightarrow \mathfrak{R}$, resulting in a function-decomposable filter that remains easy to implement, while satisfying the desired properties. For this reason, our work will also make use of PRF-wrapped Cuckoo filters, which we formally define below.

Definition 11. *Let $pp = (s, \lambda_I, \lambda_T, num)$ be a tuple of positive integers. We define an $(s, \lambda_I, \lambda_T, num)$-PRF-wrapped Cuckoo filter to be the AMQ-PDS with algorithms defined in Fig. 7, with $pp = (s, \lambda_I, \lambda_T, num)$, making use of hash functions $H_T : \mathfrak{D} \rightarrow \{0,1\}^{\lambda_T}$ and $H_I : \mathfrak{D} \rightarrow \{0,1\}^{\lambda_I}$.*

Our next step is to derive upper bounds on the NAI* false positive probability and the NAI* insertion failure probability for PRF-wrapped Cuckoo filters.

Lemma 4. *Let Π be an $(s, \lambda_I, \lambda_T, num)$-PRF-wrapped Cuckoo filter using random functions $F : \mathfrak{D} \rightarrow \mathfrak{R}$, $H_T : \mathfrak{D} \rightarrow \{0,1\}^{\lambda_T}$, and $H_I : \mathfrak{D} \rightarrow \{0,1\}^{\lambda_I}$. Then, for any n, $P^*_{\Pi,pp}(FP \,|\, n) \leq 1 - \left(1 - 2^{-\lambda_T}\right)^{2s+1} + \frac{n}{|\mathfrak{R}|}$.*

Proof (sketch). We demonstrate that, apart from the collision probability in the range of F between the queried element and those used to create the state, the false positive probability bound in [10] upper bounds the probability for any n-NAI* state. For the full proof, see Supplementary Material.

Lemma 5. *Let Π be an $(s, \lambda_I, \lambda_T, num)$-PRF-wrapped Cuckoo filter using random functions $F : \mathfrak{D} \rightarrow \mathfrak{R}$, $H_T : \mathfrak{D} \rightarrow \{0,1\}^{\lambda_T}$, and $H_I : \mathfrak{D} \rightarrow \{0,1\}^{\lambda_I}$. Then, for any n,*

$$P^*_{\Pi,pp}(IF \,|\, n) \leq \frac{2}{\left(|\mathfrak{R}| \cdot 2^{\lambda_T + \lambda_I - 1}\right)^{s-1}} \binom{n}{s} \prod_{i=1}^{s-1} \left[(|\mathfrak{R}| - i)(2^{\lambda_T} - i)\right].$$

Proof (sketch). We construct an algorithm that inserts all $X_1, ..., X_n$, using a modified insertion algorithm where an element's tag is added to both of its buckets (if they do not already contain it), and with s set to ∞. Let the resulting state be denoted by Δ. We show that the insertion failure probability of any n-NAI* state with s equal to some *limit* can be upper bounded by the probability that the load of any bucket in Δ exceeds *limit*. For the full proof, see Supplementary Material.

3.5 Consistency rules

In this work, we will consider AMQ-PDS that satisfy some properties that we refer to as consistency rules, specified below. These rules are satisfied by Counting and Cuckoo filters.

Definition 12 (AMQ-PDS consistency rules). *Consider an AMQ-PDS Π. We say Π has*

- Successful deletion of positives *if for all* $x \in \mathfrak{D}, \sigma \in \Sigma$, $\top \leftarrow \mathsf{qry}(x, \sigma) \implies (\top, \sigma') \leftarrow \mathsf{del}(x, \sigma)$.
- Unsuccessful deletion of negatives *if for all* $x \in \mathfrak{D}, \sigma \in \Sigma$, $\bot \leftarrow \mathsf{qry}(x, \sigma) \implies (\bot, \sigma) \leftarrow \mathsf{del}(x, \sigma)$.
- Unsuccessful operation invariance *if for all* $x \in \mathfrak{D}, \sigma \in \Sigma$, $(\bot, \sigma') \leftarrow \mathsf{ins}(x, \sigma) \implies \sigma' = \sigma$ *and* $(\bot, \sigma') \leftarrow \mathsf{del}(x, \sigma) \implies \sigma' = \sigma$.

In the insertion-only setting, AMQ-PDS satisfy an additional consistency rule: for all $x \in \mathfrak{D}, \sigma \in \Sigma$, $(\top, \sigma) \leftarrow \mathsf{ins}(x, \sigma) \implies \top \leftarrow \mathsf{qry}(x, \sigma)$. In other words, a membership query on an inserted element will always return $\top$, meaning that σ has no false negative elements. In a setting with both insertions and deletions, one might expect the same rule to hold as long as x has not yet been deleted from σ. In Def. 13, we define σ having no false negatives more precisely.

Definition 13 (No false negatives). *Let Π be an AMQ-PDS with public parameters pp satisfying reinsertion invariance, and let $\sigma \leftarrow \mathsf{setup}(pp)$. Let $\{z_i\}$ be the elements that are successfully inserted into σ. Let L_i be the list of successful operations on z_i, where each item in L_i is either $\mathsf{ins}(z_i, \sigma)$ or $\mathsf{del}(z_i, \sigma)$. We say σ has* no false negatives *if, for all z_i, if the last item in L_i is $\mathsf{ins}(z_i, \sigma)$, then $\top \leftarrow \mathsf{qry}(z_i, \sigma)$.*

Unfortunately, with deletions, we cannot say that σ contains no false negatives. They can arise as a result of inserting or deleting false positive elements, as we will see later. Therefore, Def. 13 is not satisfied by AMQ-PDS in general, and we will not require this from the AMQ-PDS we consider. Instead, we will analyse false negatives in our security proofs by using their relationship to false positives.

4 Adversarial Correctness

In this section, we analyse the correctness of AMQ-PDS under adversarial inputs. Our starting point is the simulation-based security definition for adversarial correctness in [13]. However, while their focus was on AMQ-PDS that only support insertions and membership queries, we are now interested in a more complex scenario with insertions, membership queries *and* deletions. As we will see, this increase in adversarial power requires tackling some new obstacles.

We derive bounds on the correctness of AMQ-PDS that satisfy function-decomposability, reinsertion invariance, and the consistency rules in Def. 12. Then, we apply our results to analyse Counting filters instantiated using a PRF,

```
Real-or-Ideal($\mathcal{A}, \mathcal{S}, \mathcal{D}, pp$)
$d \leftarrow\$ \{0,1\}$
if $d = 0$    // Real
    $K \leftarrow\$ \mathcal{K}; F \leftarrow R_K$
    $\sigma \leftarrow \mathsf{setup}(pp)$
    $out \leftarrow\$ \mathcal{A}^{\mathbf{Ins},\mathbf{Del},\mathbf{Qry}}$
else    // Ideal
    $out \leftarrow\$ \mathcal{S}(\mathcal{A}, pp)$
return $d' \leftarrow\$ \mathcal{D}(out)$
```

```
Oracle Ins($x$)
$(b, \sigma) \leftarrow\$ \mathsf{ins}^F(x, \sigma)$
return $b$

Oracle Del($x$)
$(b, \sigma) \leftarrow\$ \mathsf{del}^F(x, \sigma)$
return $b$

Oracle Qry($x$)
return $\mathsf{qry}^F(x, \sigma)$
```

Fig. 3: Correctness game for AMQ-PDS $\mathit{\Pi}$.

and PRF-wrapped Cuckoo filters. In both cases, we provide concrete guarantees on their adversarial correctness.

In the following, we consider an adversary $\mathcal{A}$ interacting with an AMQ-PDS $\mathit{\Pi}$ through an API, which we model as three oracles: **Ins**, which inserts elements of its choice into $\mathit{\Pi}$, **Del**, which deletes elements of its choice from $\mathit{\Pi}$, and **Qry**, which responds to membership queries (i.e. whether x has been inserted into $\mathit{\Pi}$).

4.1 Notions of Correctness

We employ a simulation-based approach to analysing the adversarial correctness of AMQ-PDS, which proceeds as follows. The adversary $\mathcal{A}$ plays in either the "real" or "ideal" world. In the real world, $\mathcal{A}$ interacts with a keyed AMQ-PDS $\mathit{\Pi}$, through oracles allowing it to make insertions, deletions and membership queries on elements of its choice. In the ideal world, $\mathcal{A}$ instead interacts with a simulator $\mathcal{S}$, constructed such that it provides an NAI* view of $\mathit{\Pi}$ to $\mathcal{A}$. (Note that this differs from the definition of adversarial correctness in [13], which required $\mathcal{S}$ to provide an NAI view.)

$\mathcal{A}$ then produces some arbitrary output, which the distinguisher $\mathcal{D}$ uses to compute which world $\mathcal{A}$ was operating in. Finally, we bound $\mathcal{D}$'s ability to distinguish between the two worlds. This allows us to quantify $\mathcal{A}$'s probability of achieving any adversarial goal in the real world (through adaptive insertions, deletions and membership queries) by relating it to the ideal world, which we know how to analyse.

In Fig. 3, we define the Real-or-Ideal game.

Definition 14. *Let $\mathit{\Pi}$ be an AMQ-PDS with public parameters pp, and let R_K be a keyed function family. We say $\mathit{\Pi}$ is $(q_{ins}, q_{qry}, q_{del}, t_a, t_d, t_s, \varepsilon)$-adversarially correct if, for all adversaries $\mathcal{A}$ running in time at most t_a and making q_{ins}, q_{qry}, q_{del} queries to oracles **Ins**,**Qry**,**Del** respectively in the Real-or-Ideal game (Fig. 3) with a simulator $\mathcal{S}$* that provides an NAI* view of $\mathit{\Pi}$ to $\mathcal{A}$ *and runs in time at most t_s, and for all distinguishers $\mathcal{D}$ running in time at most t_d, we have:*

$$Adv^{RoI}_{\mathit{\Pi},\mathcal{A},\mathcal{S}}(\mathcal{D}) := \left| \Pr[\mathit{Real}(\mathcal{A},\mathcal{D}) = 1] - \Pr[\mathit{Ideal}(\mathcal{A},\mathcal{D},\mathcal{S}) = 1] \right| \leq \varepsilon.$$

Remark 2. We discuss why Def. 14 captures adversarial correctness, by outlining how it can be used to analyse a specific adversarial goal. Consider an adversary $\mathcal{A}$ that, throughout its execution, makes **Ins** and **Del** queries on adversarially selected inputs $x_1, ..., x_n$, interspersed with **Qry** queries, and ending with a final membership query $\mathbf{Qry}(x)$ with $x \leftarrow\!\!\$\, \mathfrak{D} \setminus \{x_1, ..., x_n\}$. Suppose the output of $\mathcal{A}$ is the result of that final query, and $\mathcal{D}$'s output is identical to that of $\mathcal{A}$. Then, $\Pr[\mathit{Real}(\mathcal{A}, \mathcal{D})]$ is the adversarial false positive probability of Π produced by $\mathcal{A}$, for which we cannot directly compute an upper bound, since $\mathcal{A}$ makes adaptive queries. However, $\Pr[\mathit{Ideal}(\mathcal{A}, \mathcal{D}, \mathcal{S})]$ is the NAI* false positive probability, for which we can derive upper bounds for our AMQ-PDS of interest. Then, if Def. 14 is satisfied, it means we can upper bound $\Pr[\mathit{Real}(\mathcal{A}, \mathcal{D})]$ by $\Pr[\mathit{Ideal}(\mathcal{A}, \mathcal{D}, \mathcal{S})]+\varepsilon$. Note that our definition covers any adversarial goal (see [13, Appendix C.2]).

In Fig. 4, we construct a simulator $\mathcal{S}$ providing an NAI* view for function-decomposable AMQ-PDS supporting insertions, deletions and membership queries. We first observe that the state constructed by $\mathcal{S}$ is always an NAI* state (Def. 7). Every insertion of element z_i either executes $\mathsf{ins}^{\mathrm{Id}_{\mathfrak{R}}}(\cdot, \sigma)$ on fresh $X_i \leftarrow\!\!\$\, \mathfrak{R}$ or does not change the state if on a currently inserted element. Moreover, only deletions of z_i that are currently inserted run $\mathsf{del}^{\mathrm{Id}_{\mathfrak{R}}}(X_i, \sigma)$. Further, by inspection, the runtime of $\mathcal{S}$ is not significantly higher than that of the underlying AMQ-PDS.

Theorem 1. *Let $q_{ins}, q_{del}, q_{qry}$ be non-negative integers, and let t_a, $t_d > 0$. Let $F\colon \mathfrak{D} \to \mathfrak{R}$. Let Π be an AMQ-PDS with public parameters pp and oracle access to F, such that Π satisfies the consistency rules from Def. 12, F-decomposability (Def. 2), and reinsertion invariance (Def. 3). Let α, β, γ be the number of calls to F required to insert, query, delete an element respectively in Π using its* $\mathsf{ins}, \mathsf{qry}, \mathsf{del}$ *algorithms.*

If $R_K\colon \mathfrak{D} \to \mathfrak{R}$ is an $(\alpha q_{ins} + \beta q_{qry} + \gamma q_{del}, t_a + t_d, \varepsilon)$-secure pseudorandom function with key $K \leftarrow\!\!\$\, \mathcal{K}$, then Π is $(q_{ins}, q_{qry}, q_{del}, t_a, t_s, t_d, \varepsilon')$-adversarially correct with respect to the simulator in Fig. 4, where $t_s \approx t_a$ and $\varepsilon' = \varepsilon + 2P^*_{\Pi,pp}(IF \mid q_{ins}) + (q_{ins} + 2q_{qry} + q_{del}) \cdot P^*_{\Pi,pp}(FP \mid q_{ins})$.

Proof. We define an intermediate game G^* in Fig. 5. Let *Real* denote the $d = 0$ version of Real-or-G^*, let G^* denote the $d = 1$ version of Real-or-G^* (or equivalently, the $d = 0$ version of G^*-or-Ideal), and let *Ideal* denote the $d = 1$ version of G^*-or-Ideal. Then,

$$\begin{aligned}\mathrm{Adv}^{\mathrm{RoI}}_{\Pi,\mathcal{A},\mathcal{S}}(\mathcal{D}) := & \left|\Pr\left[\,\mathit{Real}(\mathcal{A}, \mathcal{D})=1\,\right] - \Pr\left[\,\mathit{Ideal}(\mathcal{A}, \mathcal{D}, \mathcal{S})=1\,\right]\right| \\ \leq & \left|\Pr[\mathit{Real}(\mathcal{A}, \mathcal{D})=1] - \Pr[G^*(\mathcal{A}, \mathcal{D})=1]\right| \\ & + \left|\Pr[G^*(\mathcal{A}, \mathcal{D})=1] - \Pr[\mathit{Ideal}(\mathcal{A}, \mathcal{D}, \mathcal{S})=1]\right|. \qquad (1)\end{aligned}$$

Our proof proceeds by bounding the closeness of *Real*, G^* in Lemma 6 in terms of the PRF advantage, and that of G^*, *Ideal* in Lemma 7 in terms of the probability of some "bad" event. Then, we combine these lemmas to obtain our result.

```
Simulator $\mathcal{S}(\mathcal{A}, pp)$
$F \leftarrow\!\!\$\ \mathsf{Funcs}[\mathfrak{D}, \mathfrak{R}]$
$\sigma \leftarrow \mathsf{setup}(pp)$
$\sigma^* \leftarrow \mathsf{setup}(pp)$
$\text{inserted} \leftarrow \{\}$
$f \leftarrow \{\}$
return $\mathcal{A}^{\textbf{InsSim, DelSim, QrySim}}$

Oracle InsSim($x$)
$(c^{G^*}, \sigma^*) \leftarrow\!\!\$\ \mathsf{ins}^F(x, \sigma^*)$
$c^{Ideal} \leftarrow \top$
// If it's not already inserted
if $\text{inserted}[x] = \bot$
    $f[x] \leftarrow\!\!\$\ \mathfrak{R}$
    $(c^{Ideal}, \sigma) \leftarrow\!\!\$\ \mathsf{ins}^{\mathrm{Id}_\mathfrak{R}}(f[x], \sigma)$
    // If the insertion succeeded
    if $c^{Ideal} = \top$
        $\text{inserted}[x] \leftarrow \top$
return $c^{Ideal}$ $\boxed{\textbf{return } c^{G^*}}$

Oracle DelSim($x$)
$(b^{G^*}, \sigma^*) \leftarrow\!\!\$\ \mathsf{del}^F(x, \sigma^*)$
$b^{Ideal} \leftarrow \bot$
if $\text{inserted}[x] = \top$
    $b^{Ideal} \leftarrow \top$
    $\text{inserted}[x] = \bot$
    $(-, \sigma) \leftarrow\!\!\$\ \mathsf{del}^{\mathrm{Id}_\mathfrak{R}}(f[x], \sigma)$
return $b^{Ideal}$ $\boxed{\textbf{return } b^{G^*}}$

Oracle QrySim($x$)
$a^{G^*} \leftarrow \mathsf{qry}^F(x, \sigma^*)$
$a^{Ideal} \leftarrow \top$
// If it isn't inserted
if $\text{inserted}[x] = \bot$
    $a^{Ideal} \leftarrow\!\!\$\ \mathsf{qry}^{\mathrm{Id}_\mathfrak{R}}(X \leftarrow\!\!\$\ \mathfrak{R}, \sigma)$
return $a^{Ideal}$ $\boxed{\textbf{return } a^{G^*}}$
```

Fig. 4: Simulator and $\boxed{G^*}$ for AMQ-PDS with deletions.

Lemma 6. *The difference in probability of an arbitrary t_d-distinguisher $\mathcal{D}$ outputting 1 in experiments of game Real-or-G^* in Fig. 5 with a $(q_{ins}, q_{qry}, q_{del}, t_a)$-AMQ-PDS adversary $\mathcal{A}$ is bounded by the maximal PRF advantage ε of an $(\alpha q_{ins} + \beta q_{qry} + \gamma q_{del}, t_a + t_d, \varepsilon)$-PRF adversary attacking R_K:*

$$Adv^{\text{Real-or-}G^*}_{\Pi,\mathcal{A},\mathcal{S}}(\mathcal{D}) := |\Pr[\,Real(\mathcal{A},\mathcal{D}) = 1\,] - \Pr[\,G^*(\mathcal{A},\mathcal{D}) = 1\,]| \leq \varepsilon.$$

Proof. Consider the PRF adversary $\mathcal{B}$ (Fig. 1), instantiating the AMQ-PDS queried by $\mathcal{A}$ using its **RoR** oracle, in relation to the Real-or-G^* game (Fig. 5). When $b = 0$, $\mathcal{B}$ is running *Real* for $\mathcal{A}$, and when $b = 1$, $\mathcal{B}$ is instead running G^* for $\mathcal{A}$. Then, the advantage of $\mathcal{B}$ is $\mathrm{Adv}^{PRF}_R(\mathcal{B}) = \mathrm{Adv}^{\text{Real-or-}G^*}_{\Pi,\mathcal{A},\mathcal{S}}(\mathcal{D})$. Since R_K is an $(\alpha q_{\mathrm{ins}} + \beta q_{\mathrm{qry}} + \gamma q_{\mathrm{del}}, t_a + t_d, \varepsilon)$-secure PRF, $\mathrm{Adv}^{\text{Real-or-}G^*}_{\Pi,\mathcal{A},\mathcal{S}}(\mathcal{D}) \leq \varepsilon$.

Lemma 7. *The difference in probability of an arbitrary t_d-distinguisher $\mathcal{D}$ outputting 1 in experiments of game G^*-or-Ideal in Fig. 5 with a $(q_{ins}, q_{qry}, q_{del}, t_a)$-*

```
Real-or-$G^*(\mathcal{A}, \mathcal{D}, pp)$

$d \leftarrow\!\!\$\ \{0,1\}$
if $d = 0$   // Real
    $K \leftarrow\!\!\$\ \mathcal{K}; F \leftarrow R_K$
else   // $G^*$
    $F \leftarrow\!\!\$\ \mathsf{Funcs}[\mathfrak{D}, \mathfrak{R}]$
$\sigma \leftarrow \mathsf{setup}(pp)$
$out \leftarrow\!\!\$\ \mathcal{A}^{\mathbf{Ins},\mathbf{Del},\mathbf{Qry}}$
return $d' \leftarrow\!\!\$\ \mathcal{D}(out)$
```

```
$G^*$-or-Ideal$(\mathcal{A}, \mathcal{S}, \mathcal{D}, pp)$

$d \leftarrow\!\!\$\ \{0,1\}$
if $d = 0$   // $G^*$
    $F \leftarrow\!\!\$\ \mathsf{Funcs}[\mathfrak{D}, \mathfrak{R}]$
    $\sigma \leftarrow \mathsf{setup}(pp)$
    $out \leftarrow\!\!\$\ \mathcal{A}^{\mathbf{Ins},\mathbf{Del},\mathbf{Qry}}$
else   // Ideal
    $out \leftarrow\!\!\$\ \mathcal{S}(\mathcal{A}, pp)$
return $d' \leftarrow\!\!\$\ \mathcal{D}(out)$
```

Fig. 5: Intermediate game G^* for the proof of Theorem 1.

AMQ-PDS adversary $\mathcal{A}$ is bounded as follows:

$$\begin{aligned} Adv^{G^*\text{-}or\text{-}Ideal}_{\Pi,\mathcal{A},\mathcal{S}}(\mathcal{D}) &:= |\Pr[\, G^*(\mathcal{A},\mathcal{D}) = 1\,] - \Pr[\, Ideal(\mathcal{A},\mathcal{D},\mathcal{S}) = 1\,]| \\ &\leq 2P^*_{\Pi,pp}(IF \mid q_{ins}) + (q_{ins} + 2q_{qry} + q_{del}) \cdot P^*_{\Pi,pp}(FP \mid q_{ins}). \end{aligned}$$

Proof. We wish to bound the probability of distinguishing between G^* and *Ideal*. Let $\mathbf{E}$ be the divergence event between G^* and *Ideal*, which occurs due to a mismatch in responses to $\mathbf{Qry}, \mathbf{Del}, \mathbf{Ins}$ queries across the two games (see Fig. 4).

First, we observe that *Ideal* cannot have false negative responses to membership queries, but G^* could have. If $\mathcal{A}$ induces a false negative for some element x in G^* and then calls $\mathbf{Qry}(x)$, the two games would diverge with probability one. False negatives lead to repercussions when comparing responses to all types of queries across the two games. Therefore, we will deal with them separately, by defining $\mathbf{E}_{\text{FN}}$ to be the event that a false negative occurs in G^* before any other query response mismatch. We then split the analysis of event $\mathbf{E}$ into two parts: (1) the query response mismatch occurs without a false negative occurring in G^* beforehand (i.e.$\neg\mathbf{E}_{\text{FN}}$), or (2) the query response mismatch occurs with a false negative occurring in G^* beforehand (i.e. $\mathbf{E}_{\text{FN}}$). Then,

$$\Pr[\,\mathbf{E}\,] \leq \Pr[\,\mathbf{E} \wedge \neg\mathbf{E}_{\text{FN}}\,] + \Pr[\,\mathbf{E} \wedge \mathbf{E}_{\text{FN}}\,] \leq \Pr[\,\mathbf{E} \wedge \neg\mathbf{E}_{\text{FN}}\,] + \Pr[\,\mathbf{E}_{\text{FN}}\,].$$

We will analyse $\mathbf{E} \wedge \neg\mathbf{E}_{\text{FN}}$ for each query type separately. Let a^G_i, b^G_i, c^G_i denote the responses to $\mathcal{A}$'s i-th query, deletion, and insertion query in game $G \in \{G^*, Ideal\}$. Then, the games diverge the first time that $a^{G^*}_i, b^{G^*}_i$ or $c^{G^*}_i$ does not match a_i^{Ideal}, b_i^{Ideal} or c_i^{Ideal}, respectively. We define

$$\mathbf{E}_{\mathbf{Qry}} := \left[\begin{bmatrix}\text{The first query response mismatch is}\\ a_i^{Ideal} \neq a_i^{G^*} \text{ for some } i \in [q_{\text{qry}}]\end{bmatrix} \wedge \neg \mathbf{E}_{\text{FN}}\right], \tag{2}$$

$$\mathbf{E}_{\mathbf{Del}} := \left[\begin{bmatrix}\text{The first query response mismatch is}\\ b_i^{Ideal} \neq b_i^{G^*} \text{ for some } i \in [q_{\text{del}}]\end{bmatrix} \wedge \neg \mathbf{E}_{\text{FN}}\right], \tag{3}$$

$$\mathbf{E}_{\mathbf{Ins}} := \left[\begin{bmatrix}\text{The first query response mismatch is}\\ c_i^{Ideal} \neq c_i^{G^*} \text{ for some } i \in [q_{\text{ins}}]\end{bmatrix} \wedge \neg \mathbf{E}_{\text{FN}}\right]. \tag{4}$$

Hence,

$$\Pr[\mathbf{E}] \leq \Pr[\mathbf{E}_{\text{FN}}] + \Pr[\mathbf{E}_{\mathbf{Qry}}] + \Pr[\mathbf{E}_{\mathbf{Del}}] + \Pr[\mathbf{E}_{\mathbf{Ins}}]. \tag{5}$$

We now proceed to bound the probability of each event in Eq. (5). In the following, we take the probability over the randomness used by $\mathcal{A}$ (which we refer to as $\mathcal{A}$'s coins), and the randomness used by game $G \in \{G^*, Ideal\}$ to answer $\mathcal{A}$'s queries (which we refer to as G's coins). We will use x_i, y_i and z_i to denote the input to $\mathcal{A}$'s i-th query, deletion and insertion query, respectively.

Calculation of $\Pr[\mathbf{E}_{FN}]$. We start by analysing the probability of a false negative occurring in G^*. Our key observation is that false negatives can only occur from inserting or deleting false positives.

Consider an element x that is a false positive due to insertions of $\{z_1, ..., z_\ell\}$, where $\ell \geq 1$. By the consistency rule *successful deletion of positives*, the deletion of x will succeed, although it was never inserted. However, this may cause elements in $\{z_1, ..., z_\ell\}$ to become false negatives. Now, consider inserting this false positive x. By *reinsertion invariance*, the state will remain unchanged, but x will become a true positive. Then, deleting any element in $\{x, z_1, ..., z_\ell\}$ will succeed, but may cause other elements in $\{x, z_1, ..., z_\ell\}$ to become false negatives.

Recall that we are interested in analysing the probability of a false negative occurring in G^* *before* any other mismatch in query responses across the two games. Therefore, we do not need to consider deletions of false positives; it would result in a mismatch in **Del** responses, since *Ideal* does not allow deletions of false positives while G^* does. We will then focus solely on false negatives caused by insertions of false positives in the following. We write

$$\begin{aligned}\Pr[\mathbf{E}_{\text{FN}}] &:= \Pr\begin{bmatrix}\text{A false negative occurs in } G^*\\ \text{before a query response mismatch occurred}\end{bmatrix}\\ &\leq \Pr\begin{bmatrix}\text{A false positive is inserted in } G^*\\ \text{before a query response mismatch occurred}\end{bmatrix}\\ &\leq \sum_{i=1}^{q_{\text{ins}}} \Pr\begin{bmatrix}z_i \text{ is the first false positive inserted in } G^*\\ \text{before a query response mismatch occurred}\end{bmatrix}.\end{aligned}$$

Let σ_i^* denote the state of $\mathit{\Pi}$ in game G^* just before the i-th **Ins** query. Then, since no prior query response mismatch occurred and z_i is the first false positive inserted, σ_i^* contains no false negatives up to this point. Then,

$$\begin{aligned}\Pr[\mathbf{E}_{\mathrm{FN}}] &\leq \sum_{i=1}^{q_{\mathrm{ins}}} \Pr\left[\begin{array}{c}[z_i \text{ is a false positive in } \sigma_i^*]\wedge \\ [\sigma_i^* \text{ has no false negatives}]\wedge \\ [\text{no prior query response mismatch occurred}]\end{array}\right] \\ &\leq \sum_{i=1}^{q_{\mathrm{ins}}} \Pr_{\substack{G^*\text{'s coins} \\ \mathit{Ideal}\text{'s coins} \\ \mathcal{A}\text{'s coins}}}\left[\begin{array}{c}[\text{inserted}[z_i] = \bot] \wedge [\top \leftarrow\$ \, \mathsf{qry}^F(z_i, \sigma_i^*)]\wedge \\ [\sigma_i^* \text{ has no false negatives}]\wedge \\ [\text{no prior query response mismatch occurred}]\end{array}\right]. \quad (6)\end{aligned}$$

Let L_i be the list of successful operations on σ^* in G^* up to the i-th **Ins** query, where each item in L_i is either $\mathsf{ins}^F(\cdot, \sigma^*)$ or $\mathsf{del}^F(\cdot, \sigma^*)$ on $z_1, ..., z_{i-1}$. By the consistency rule *unsuccessful operation invariance*, we do not need to consider unsuccessful operations when constructing σ_i^*. So,

$$\Pr[\mathbf{E}_{\mathrm{FN}}] \leq \sum_{i=1}^{q_{\mathrm{ins}}} \Pr_{\substack{G^*\text{'s coins} \\ \mathit{Ideal}\text{'s coins} \\ \mathcal{A}\text{'s coins}}}\left[\begin{array}{c}\sigma_i^* \leftarrow \mathsf{setup}(pp) \\ \textbf{for } \mathrm{op}^F(z_j, \sigma^*) \in L_i : (\top, \sigma_i^*) \leftarrow \mathrm{op}^F(z_j, \sigma_i^*): \\ [\text{inserted}[z_i] = \bot] \wedge [\top \leftarrow\$ \, \mathsf{qry}^F(z_i, \sigma_i^*)]\wedge \\ [\sigma_i^* \text{ has no false negatives}]\wedge \\ [\text{no prior query response mismatch occurred}]\end{array}\right]. \quad (7)$$

Now, if no prior query response mismatch has occurred, inserted$[z_i] = \bot$ implies that either z_i was never inserted into σ_i^*, or z_i was inserted but then deleted. In the latter scenario, since σ_i^* contains no false negatives up to this point (as per Eq. (7)), z_i must be a positive at the time of its deletion. Then, by the consistency rule *successful deletion of positives*, its deletion will succeed, thus fully undoing the effect of its insertion on σ_i^*. Since F is a random function satisfying F-decomposability and $\mathcal{A}$ has no information about F, we write Eq. (7) as

$$\Pr\left[\,\mathbf{E}_{\mathrm{FN}}\,\right] \leq \sum_{i=1}^{q_{\mathrm{ins}}} \Pr_{\substack{G^*\text{'s coins} \\ \mathit{Ideal}\text{'s coins} \\ \mathcal{A}\text{'s coins}}}\left[\begin{array}{c}\sigma_i^* \leftarrow \mathsf{setup}(pp) \\ \textbf{for } \mathrm{op}^F(z_j, \sigma^*) \in L_i : (\top, \sigma_i^*) \leftarrow \mathrm{op}^{\mathrm{Id}_{\mathfrak{R}}}(F(z_j), \sigma_i^*) : \\ [\top \leftarrow\$ \, \mathsf{qry}^{\mathrm{Id}_{\mathfrak{R}}}(X \leftarrow\$ \, \mathfrak{R}, \sigma_i^*)]\wedge \\ [\sigma_i^* \text{ has no false negatives}]\wedge \\ [\text{no prior query response mismatch occurred}]\end{array}\right].$$

Now, since F is a random function and $\mathcal{A}$ has no information about F, we can replace every first insertion of an element $F(z_j)$ by $X_{z_j} \leftarrow\$ \, \mathfrak{R}$ (i.e. σ_i^* satisfies *insertion unpredictability*). For repeated insertions on an element, we have two possibilities. If this element has not been deleted since its last insertion, the repeated insertion will not change the state, due to *reinsertion invariance*. However, if it has been deleted since its last insertion, it will change the state in the same way as its first insertion, since both use the same F. Therefore, we can rewrite the above by sampling $|\{z_1, ..., z_{i-1}\}|$ random strings, and associating each string to a distinct z_j, giving

$$\Pr[\mathbf{E}_{\mathrm{FN}}] \leq \sum_{i=1}^{q_{\mathrm{ins}}} \Pr_{\substack{\mathit{Ideal}\text{'s coins}\\ \mathcal{A}\text{'s coins}}} \left[\begin{array}{c} \ell \leftarrow |\{z_1, ..., z_{i-1}\}| \\ \{u_1, ..., u_\ell\} \leftarrow \{z_1, ..., z_{i-1}\} \\ X_{u_1}, ..., X_{u_\ell} \leftarrow\!\!\$\, \mathfrak{R} \\ \sigma_i^* \leftarrow \mathsf{setup}(pp) \\ \textbf{for } \mathrm{op}^F(z_j, \sigma^*) \in L_i{:}(\top, \sigma_i^*) {\leftarrow} \mathrm{op}^{\mathrm{Id}_{\mathfrak{R}}}(X_{z_j}, \sigma_i^*){:} \\ [\top \leftarrow\!\!\$\, \mathsf{qry}^{\mathrm{Id}_{\mathfrak{R}}}(X \leftarrow\!\!\$\, \mathfrak{R}, \sigma_i^*)] \wedge \\ [\sigma_i^* \text{ has no false negatives}] \wedge \\ [\text{no prior query response mismatch occurred}] \end{array} \right]. \quad (8)$$

We now argue that every σ_i^* is an n-NAI* state, where n is upper bounded by q_{ins}, by showing that it satisfies the requirements in Corollary 1. Firstly, observe that the construction of σ_i^* in Eq. (8) enforces insertion unpredictability (Def. 6). Secondly, there are at most q_{ins} insertions in σ_i^*. Thirdly, since no query response mismatch has yet occurred, all deletions must be on elements for which the preceding successful operation was an insertion. Finally, since there are no false negatives up to this point and *reinsertion invariance* holds, any insertion on a currently inserted element will not change the state.

Let $\mathcal{U}_{\Pi,pp}^{\mathrm{Id}_{\mathfrak{R}}}$ be the algorithm from Def. 8 that, given $X_1, ..., X_n$, outputs an NAI* state created using $\mathsf{ins}^{\mathrm{Id}_{\mathfrak{R}}}, \mathsf{del}^{\mathrm{Id}_{\mathfrak{R}}}$ on $X_1, ..., X_n$ with the maximal false positive probability. Then, no matter how σ_i^* is created, the state output by $\mathcal{U}_{\Pi,pp}^{\mathrm{Id}_{\mathfrak{R}}}$ will result in an equal or higher false positive probability than that of σ_i^*. Since $\ell \leq q_{\mathsf{ins}}$ and with more distinct insertions, $\mathcal{U}_{\Pi,pp}^{\mathrm{Id}_{\mathfrak{R}}}$ may be able to create a state with even higher false positive probability,

$$\Pr[\,\mathbf{E}_{\mathrm{FN}}\,] \leq \sum_{i=1}^{q_{\mathrm{ins}}} \Pr_{\mathit{Ideal}\text{'s coins}} \left[\begin{array}{c} X_1, ..., X_{q_{\mathrm{ins}}} \leftarrow\!\!\$\, \mathfrak{R} \\ \sigma \leftarrow \mathcal{U}_{\Pi,pp}^{\mathrm{Id}_{\mathfrak{R}}}(X_1, ..., X_{q_{\mathrm{ins}}}) : \\ \top \leftarrow\!\!\$\, \mathsf{qry}^{\mathrm{Id}_{\mathfrak{R}}}(X \leftarrow\!\!\$\, \mathfrak{R}, \sigma) \end{array} \right].$$

Finally, applying Def. 8, we obtain

$$\Pr[\,\mathbf{E}_{\mathrm{FN}}\,] \leq q_{\mathrm{ins}} \cdot P_{\Pi,pp}^*(FP \,|\, q_{\mathrm{ins}}). \quad (9)$$

Calculation of $\Pr[\,\mathbf{E}_{\boldsymbol{Qry}}\,]$. We first rewrite Eq. (2) using the union bound as

$$\Pr[\mathbf{E}_{\mathbf{Qry}}] \leq \sum_{i=1}^{q_{\mathrm{qry}}} \Pr\left[\left[\begin{array}{c} [\mathbf{Qry}(x_i) \text{ yields the first mismatch}] \wedge \\ [[(a_i^{\mathit{Ideal}} = \top) \wedge (a_i^{G^*} = \bot)] \\ \vee [(a_i^{\mathit{Ideal}} = \bot) \wedge (a_i^{G^*} = \top)]] \end{array} \right] \wedge \neg \mathbf{E}_{\mathrm{FN}} \right]. \quad (10)$$

We start by inspecting the **Qry** algorithms of G^* and *Ideal* to see where they could diverge. In G^*, the responses to $\mathcal{A}$'s **Qry** queries are always computed using the same function F, while in *Ideal*, a fresh random string $X \leftarrow\!\!\$\, \mathfrak{R}$ is sampled each time a non-inserted element is queried.

Let σ_i denote the state of Π in game *Ideal* just before the i-th **Qry** query, and σ_i^* denote the corresponding state in game G^*. Since $\mathbf{Qry}(x_i)$ yields the *first* query response mismatch, both G^* and *Ideal* must contain the same set of inserted elements. As $\mathbf{E}_{FN}$ did not yet occur, σ_i^* has no false negatives. Moreover, **Qry** queries in *Ideal* do not give false negative responses. This means that **Qry** queries on elements that were inserted (and not yet deleted) will always return a positive response in both games. Therefore, in order for x_i to yield a mismatch in **Qry** query responses between the games, we must have that x_i is not currently inserted in *Ideal* (i.e. inserted$[x_i] = \bot$ in line 4 of **QrySim**). This gives

$$\begin{aligned}\Pr\left[\mathbf{E}_{\mathbf{Qry}}\right] \leq \sum_{i=1}^{q_{\mathrm{qry}}} \Bigg[\Pr\left[\begin{bmatrix}[\mathbf{Qry}(x_i) \text{ yields the first mismatch }]\wedge \\ [\text{inserted}[x_i] = \bot] \wedge [a_i^{Ideal} = \top]\end{bmatrix} \wedge \neg\mathbf{E}_{\mathrm{FN}}\right] \\ + \Pr\left[\begin{bmatrix}[\mathbf{Qry}(x_i) \text{ yields the first mismatch }]\wedge \\ [\text{inserted}[x_i] = \bot] \wedge [a_i^{G^*} = \top]\end{bmatrix} \wedge \neg\mathbf{E}_{\mathrm{FN}}\right]\Bigg] \qquad (11) \\ := \sum_{i=1}^{q_{\mathrm{qry}}} \left[\Pr\left[\mathbf{E}_{\mathbf{Qry}}^{Ideal}\right] + \Pr\left[\mathbf{E}_{\mathbf{Qry}}^{G^*}\right]\right], \qquad (12)\end{aligned}$$

where, for simplicity, we will use $\Pr\left[\mathbf{E}_{\mathbf{Qry}}^{Ideal}\right]$ to denote the first term of Eq. (11), and $\Pr\left[\mathbf{E}_{\mathbf{Qry}}^{G^*}\right]$ to denote the second term.

We start by bounding $\Pr\left[\mathbf{E}_{\mathbf{Qry}}^{Ideal}\right]$. In *Ideal*, a fresh random string $X \leftarrow\!\!\$\, \mathfrak{R}$ is sampled each time a non-inserted element is queried, and so

$$\Pr\left[\mathbf{E}_{\mathbf{Qry}}^{Ideal}\right] \leq \Pr_{\substack{Ideal\text{'s coins} \\ \mathcal{A}\text{'s coins}}}\begin{bmatrix}[\mathbf{Qry}(x_i) \text{ yields the first mismatch }]\wedge \\ [\top \leftarrow\!\!\$\, \mathsf{qry}^{\mathrm{Id}_{\mathfrak{R}}}(X \leftarrow\!\!\$\, \mathfrak{R}, \sigma_i)]\end{bmatrix}.$$

We now argue that every σ_i is an n-NAI* state, with n being upper bounded by q_{ins}, by showing that it satisfies the requirements in Def. 7. Firstly, from line 3 of **DelSim**, we observe that only deletions of currently inserted elements run $\mathsf{del}^{\mathrm{Id}_{\mathfrak{R}}}(\cdot, \sigma)$, possibly changing the state. Secondly, we note that in **InsSim**, every insertion either executes $\mathsf{ins}^{\mathrm{Id}_{\mathfrak{R}}}(\cdot, \sigma)$ on $X_i \leftarrow\!\!\$\, \mathfrak{R}$, or does not change the state if it is on a currently inserted element. Therefore, σ_i is an NAI* state containing at most q_{ins} elements. Then, we can upper bound the false positive probability of σ_i by that of the NAI* state with the maximal false positive probability (Def. 8), giving $\Pr\left[\mathbf{E}_{\mathbf{Qry}}^{Ideal}\right] \leq P^*_{\Pi,pp}(FP \mid q_{\mathrm{ins}})$.

We use a reasoning similar to calculating $\mathbf{E}_{FN}$ to compute $\Pr\left[\mathbf{E}_{\mathbf{Qry}}^{G^*}\right]$, replacing z_i with x_i. Under $\neg\mathbf{E}_{FN}$, the state σ_i^* contains no false negatives. Therefore, we can apply Eq. (6) from the $\mathbf{E}_{FN}$ calculation to get

$$\begin{aligned}\Pr\left[\mathbf{E}_{\mathbf{Qry}}^{G^*}\right] &= \Pr_{\substack{G^*\text{'s coins} \\ Ideal\text{'s coins} \\ \mathcal{A}\text{'s coins}}}\left[\begin{bmatrix}[\mathbf{Qry}(x_i) \text{ yields the first mismatch }]\wedge \\ [\text{inserted}[x_i] = \bot] \wedge [\top \leftarrow\!\!\$\, \mathsf{qry}^F(x_i, \sigma_i^*)]\end{bmatrix} \wedge \neg\mathbf{E}_{\mathrm{FN}}\right] \\ &\leq \Pr_{\substack{G^*\text{'s coins} \\ Ideal\text{'s coins} \\ \mathcal{A}\text{'s coins}}}\begin{bmatrix}[\text{inserted}[x_i] = \bot] \wedge [\top \leftarrow\!\!\$\, \mathsf{qry}^F(x_i, \sigma_i^*)]\wedge \\ [\sigma_i^* \text{ has no false negatives}]\wedge \\ [\text{no prior query response mismatch occurred}]\end{bmatrix} \\ &\leq P^*_{\Pi,pp}(FP \mid q_{\mathrm{ins}}). \qquad (13)\end{aligned}$$

Substituting $\Pr\left[\mathbf{E}^{Ideal}_{\mathbf{Qry}}\right], \Pr\left[\mathbf{E}^{G^*}_{\mathbf{Qry}}\right]$ in Eq. (12) gives

$$\Pr\left[\,\mathbf{E}_{\mathbf{Qry}}\,\right] \le \sum_{i=1}^{q_{\text{qry}}} 2P^*_{\Pi,pp}(FP \mid q_{\text{ins}}) = 2q_{\text{qry}} \cdot P^*_{\Pi,pp}(FP \mid q_{\text{ins}}). \tag{14}$$

Calculation of $\Pr\left[\,\mathbf{E}_{\boldsymbol{Del}}\,\right]$. We first rewrite Eq. (3) using the union bound as

$$\Pr\left[\,\mathbf{E}_{\mathbf{Del}}\,\right] \le \sum_{i=1}^{q_{\text{del}}} \Pr\left[\begin{bmatrix}[\mathbf{Del}(y_i) \text{ yields the first mismatch}]\wedge \\ [(b_i^{Ideal} = \top) \wedge (b_i^{G^*} = \bot)] \\ \vee[(b_i^{Ideal} = \bot) \wedge (b_i^{G^*} = \top)]\end{bmatrix} \wedge \neg\mathbf{E}_{\text{FN}}\right]. \tag{15}$$

We now examine the **Del** algorithms of G^* and *Ideal*. In G^*, the responses to $\mathcal{A}$'s **Del** queries are always computed using the same function F. In *Ideal*, deletions are only allowed on an element y_i if it is currently inserted in the filter, and use the same random string $f[y_i]$ that was used for y_i's insertion.

We note that in Eq. (15), we are only interested in the case where $\mathbf{Del}(y_i)$ is the first query response mismatch. In this case, both G^* and *Ideal* must contain the same set of inserted elements. As $\mathbf{E}_{FN}$ did not occur, every inserted element is a true positive in both games. We observe that in *Ideal*, true positives are always successfully deleted (see line 3 of **DelSim**), while in G^*, successful deletion of true positives is ensured by the consistency rule *successful deletion of positives*. However, by the same consistency rule, deletions of false positives also succeed in G^*, while they do not in *Ideal*. Consequently, deletions in G^* succeed on at least the elements on which they succeed in *Ideal*. Thus, it never happens that a deletion succeeds in *Ideal* but not in G^*, and we can rewrite Eq. (15) as

$$\Pr\left[\,\mathbf{E}_{\mathbf{Del}}\,\right] \le \sum_{i=1}^{q_{\text{del}}} \Pr\left[\begin{bmatrix}[\mathbf{Del}(y_i) \text{ yields the first mismatch}]\wedge \\ [(b_i^{Ideal} = \bot) \wedge (b_i^{G^*} = \top)]\end{bmatrix} \wedge \neg\mathbf{E}_{\text{FN}}\right].$$

Let σ_i^* denote the state of Π in game G^* just before the i-th **Del** query. By the consistency rule *unsuccessful deletion of negatives*,

$$\Pr\left[\,\mathbf{E}_{\mathbf{Del}}\,\right] \le \sum_{i=1}^{q_{\text{del}}} \Pr_{\substack{G^*\text{'s coins}\\ Ideal\text{'s coins}\\ \mathcal{A}\text{'s coins}}}\left[\begin{bmatrix}[\mathbf{Del}(y_i) \text{ yields the first mismatch }]\wedge \\ [\text{inserted}[y_i] = \bot] \wedge \left[\top \leftarrow\!\!\$\; \mathsf{qry}^F(y_i, \sigma_i^*)\right]\end{bmatrix} \wedge \neg\mathbf{E}_{\text{FN}}\right].$$

We use a reasoning similar to calculating $\mathbf{E}_{FN}$ to compute this, replacing z_i with y_i. Under $\neg\mathbf{E}_{FN}$, the state σ_i^* contains no false negatives. Therefore, we can apply Eq. (6) from the $\mathbf{E}_{FN}$ calculation to get

$$\begin{aligned}\Pr\left[\,\mathbf{E}_{\mathbf{Del}}\,\right] &\le \sum_{i=1}^{q_{\text{del}}} \Pr_{\substack{G^*\text{'s coins}\\ Ideal\text{'s coins}\\ \mathcal{A}\text{'s coins}}}\begin{bmatrix}[\text{inserted}[y_i] = \bot] \wedge [\top \leftarrow\!\!\$\; \mathsf{qry}^F(y_i, \sigma_i^*)]\wedge \\ [\sigma_i^* \text{ has no false negatives}]\wedge \\ [\text{no prior query response mismatch occurred}]\end{bmatrix} \\ &\le \sum_{i=1}^{q_{\text{del}}} P^*_{\Pi,pp}(FP \mid q_{\text{ins}}) = q_{\text{del}} \cdot P^*_{\Pi,pp}(FP \mid q_{\text{ins}}).\end{aligned} \tag{16}$$

Calculation of $\Pr[\mathbf{E}_{\mathbf{Ins}}]$. We first rewrite Eq. (4) as

$$\Pr[\mathbf{E}_{\mathbf{Ins}}] = \Pr\left[\begin{bmatrix}[\mathbf{Ins}(z_i) \text{ yields the first mismatch}]\wedge \\ [[(c_i^{Ideal} = \bot) \wedge (c_i^{G^*} = \top)] \\ \vee[(c_i^{Ideal} = \top) \wedge (c_i^{G^*} = \bot)]] \\ \text{for some } i \in [q_{\text{ins}}]\end{bmatrix} \wedge \neg\mathbf{E}_{\text{FN}}\right]. \tag{17}$$

Let us now compare the **Ins** algorithms of G^* and *Ideal*. In G^*, the responses to $\mathcal{A}$'s **Ins** queries are always computed using the same function F. On the other hand, in *Ideal*, a fresh random string $f[z_i] \leftarrow\!\$\, \mathfrak{R}$ is sampled at each insertion of an element z_i which is not already inserted.

Let σ_i denote the state of Π in game *Ideal* just before the i-th **Ins** query, and σ_i^* denote the corresponding state in game G^*. If $\mathbf{Ins}(z_i)$ is the first mismatch, it must be that both G^* and *Ideal* contain the same set of inserted elements up to this point. In *Ideal*, by inspecting **InsSim** we observe that the insertion of any currently inserted element z_i will always succeed. In G^*, since we are only considering the case where $\mathbf{E}_{FN}$ did not yet occur, σ_i^* has no false negatives. This means that any element z_i that was inserted and not yet deleted will result in $\top \leftarrow \mathsf{qry}^F(z_i, \sigma_i^*)$. Then, by *reinsertion invariance*, the insertion of z_i will succeed (but not change the state) in G^*. Therefore, for the first query response mismatch it must be that x_i is not currently inserted in *Ideal* (i.e. inserted$[z_i] = \bot$ in line 3 of **InsSim**). Then,

$$\begin{aligned}\Pr[\mathbf{E}_{\mathbf{Ins}}] &\le \Pr\left[\begin{bmatrix}[\mathbf{Ins}(z_i) \text{ yields the first mismatch }]\wedge \\ [c_i^{Ideal} = \bot] \wedge [\text{inserted}[z_i] = \bot] \\ \text{for some } i \in [q_{\text{ins}}]\end{bmatrix} \wedge \neg\mathbf{E}_{\text{FN}}\right] \\ &+ \Pr\left[\begin{bmatrix}[\mathbf{Ins}(z_i) \text{ yields the first mismatch }]\wedge \\ [c_i^{G^*} = \bot] \wedge [\text{inserted}[z_i] = \bot] \\ \text{for some } i \in [q_{\text{ins}}]\end{bmatrix} \wedge \neg\mathbf{E}_{\text{FN}}\right] \quad (18) \\ &:= \Pr\left[\mathbf{E}_{\mathbf{Ins}}^{Ideal}\right] + \Pr\left[\mathbf{E}_{\mathbf{Ins}}^{G^*}\right], \quad (19)\end{aligned}$$

where, for simplicity, we will use $\Pr\left[\mathbf{E}_{\mathbf{Ins}}^{Ideal}\right]$ to denote the first term of Eq. (18), and $\Pr\left[\mathbf{E}_{\mathbf{Ins}}^{G^*}\right]$ to denote the second term.

We start by computing $\Pr\left[\mathbf{E}_{\mathbf{Ins}}^{Ideal}\right]$. In *Ideal*, a fresh random string $X \leftarrow\!\$\, \mathfrak{R}$ is sampled each time a non-inserted element is queried, and so we can write

$$\Pr\left[\mathbf{E}_{\mathbf{Ins}}^{Ideal}\right] \le \Pr_{\substack{Ideal\text{'s coins} \\ \mathcal{A}\text{'s coins}}}\begin{bmatrix}[\mathbf{Ins}(z_i) \text{ yields the first mismatch }]\wedge \\ [(\bot, \sigma_i) \leftarrow\!\$\, \mathsf{ins}^{\mathrm{Id}_{\mathfrak{R}}}(X \leftarrow\!\$\, \mathfrak{R}, \sigma_i)] \\ \text{for some } i \in [q_{\text{ins}}]\end{bmatrix}.$$

Since every σ_i is an n-NAI* state, with n being upper bounded by q_{ins}, we can upper bound the insertion failure probability of σ_i by that of the NAI* state with the maximal insertion failure probability (Def. 9), giving $\Pr\left[\mathbf{E}_{\mathbf{Ins}}^{Ideal}\right] \le P^*_{\Pi,pp}(IF \mid q_{\text{ins}})$.

We now compute $\Pr\left[\mathbf{E}^{G^*}_{\mathbf{Ins}}\right]$. We have that

$$\Pr\left[\mathbf{E}^{G^*}_{\mathbf{Ins}}\right] \quad \leq \Pr_{\substack{G^*\text{'s coins}\\ \mathit{Ideal}\text{'s coins}\\ \mathcal{A}\text{'s coins}}}\left[\left[\begin{array}{c}[\mathbf{Ins}(z_i)\text{ yields the first mismatch}]\wedge\\ [(\bot,\sigma_i^*)\leftarrow\!\!\$\ \mathsf{ins}^F(z_i,\sigma_i^*)]\\ \text{for some } i\in[q_{\text{ins}}]\end{array}\right]\wedge\neg\mathbf{E}_{\text{FN}}\right].$$

Let L_i be the list of successful operations on σ^* in G^* up to the i-th **Ins** query, where each item in L_i is either $\mathsf{ins}^F(\cdot,\sigma^*)$ or $\mathsf{del}^F(\cdot,\sigma^*)$ on $z_1,...,z_{i-1}$. Recall that we do not need to consider unsuccessful operations when constructing σ_i^*, by the consistency rule *unsuccessful operation invariance*. Then,

$$\Pr\left[\mathbf{E}^{G^*}_{\mathbf{Ins}}\right] \leq \Pr_{\substack{G^*\text{'s coins}\\ \mathit{Ideal}\text{'s coins}\\ \mathcal{A}\text{'s coins}}}\left[\begin{array}{c}\sigma_i^*\leftarrow\mathsf{setup}(pp)\\ \textbf{for }\mathrm{op}^F(z_j,\sigma^*)\in L_i:(\top,\sigma_i^*)\leftarrow\mathrm{op}^F(z_j,\sigma_i^*):\\ [\text{inserted}[z_i]=\bot]\wedge[(\bot,\sigma_i^*)\leftarrow\!\!\$\ \mathsf{ins}^F(z_i,\sigma_i^*)]\wedge\\ [\sigma_i^*\text{ has no false negatives}]\wedge\\ [\mathbf{Ins}(z_i)\text{ yields the first mismatch}]\\ \text{for some } i\in[q_{\text{ins}}]\end{array}\right]$$

$$= \Pr_{\substack{G^*\text{'s coins}\\ \mathit{Ideal}\text{'s coins}\\ \mathcal{A}\text{'s coins}}}\left[\begin{array}{c}\sigma_i^*\leftarrow\mathsf{setup}(pp)\\ \textbf{for }\mathrm{op}^F(z_j,\sigma^*)\in L_i:(\top,\sigma_i^*)\leftarrow\mathrm{op}^{\mathrm{Id}_{\mathfrak{R}}}(F(z_j),\sigma_i^*):\\ [\text{inserted}[z_i]=\bot]\wedge[(\bot,\sigma_i^*)\leftarrow\!\!\$\ \mathsf{ins}^{\mathrm{Id}_{\mathfrak{R}}}(F(z_i),\sigma_i^*)]\wedge\\ [\sigma_i^*\text{ has no false negatives}]\wedge\\ [\mathbf{Ins}(z_i)\text{ yields the first mismatch}]\\ \text{for some } i\in[q_{\text{ins}}]\end{array}\right],$$

by F-decomposability. Now, since F is a random function and $\mathcal{A}$ has no information about F, we can then proceed in a similar manner as in the $\mathbf{E}_{FN}$ calculation, with the caveat that we are now interested in *any* of the q_{ins} insertions failing:

$$\Pr[\mathbf{E}^{G^*}_{\mathbf{Ins}}]\leq\Pr_{\substack{\mathit{Ideal}\text{'s coins}\\ \mathcal{A}\text{'s coins}}}\left[\begin{array}{c}\ell\leftarrow|\{z_1,...,z_{q_{\text{ins}}}\}|\\ \{u_1,...,u_\ell\}\leftarrow\{z_1,...,z_{q_{\text{ins}}}\}\\ X_{u_1},...,X_{u_\ell}\leftarrow\!\!\$\ \mathfrak{R}\\ \text{for some } i\in[q_{\text{ins}}],\\ \sigma_i^*\leftarrow\mathsf{setup}(pp)\\ \textbf{for }\mathrm{op}^F(z_j,\sigma^*)\in L_i:(\top,\sigma_i^*)\leftarrow\mathrm{op}^{\mathrm{Id}_{\mathfrak{R}}}(X_{z_j},\sigma_i^*):\\ [(\bot,\sigma_i^*)\leftarrow\!\!\$\ \mathsf{ins}^{\mathrm{Id}_{\mathfrak{R}}}(X_{z_i}\leftarrow\!\!\$\ \mathfrak{R},\sigma_i^*)]\wedge\\ [\sigma_i^*\text{ has no false negatives}]\wedge\\ [\mathbf{Ins}(z_i)\text{ yields the first mismatch}]\end{array}\right]. \quad (20)$$

Similarly as in Eq. (8), we conclude that σ_i^* in Eq. (20) is an NAI* state. Then, we again upper bound the insertion failure probability of σ_i^* by that of the NAI* state with the maximal insertion failure probability (Def. 9), giving $\Pr\left[\mathbf{E}^{G^*}_{\mathbf{Ins}}\right]\leq P^*_{\Pi,pp}(IF\,|\,q_{\text{ins}})$. Substituting $\Pr\left[\mathbf{E}^{\mathit{Ideal}}_{\mathbf{Ins}}\right],\Pr\left[\mathbf{E}^{G^*}_{\mathbf{Ins}}\right]$ in Eq. (19), we obtain

$$\Pr\left[\mathbf{E}_{\mathbf{Ins}}\right]\leq 2P^*_{\Pi,pp}(IF\,|\,q_{\text{ins}}). \quad (21)$$

Finally, substituting Eqs. (9, 14, 16, 21) in Eq. (5), we have

$$\mathrm{Adv}^{G^*\text{-or-}\mathit{Ideal}}_{\Pi,\mathcal{A},\mathcal{S}}(\mathcal{D})\leq 2P^*_{\Pi,pp}(IF\,|\,q_{\text{ins}})+(q_{\text{ins}}+2q_{\text{qry}}+q_{\text{del}})\cdot P^*_{\Pi,pp}(FP\,|\,q_{\text{ins}}).$$

To prove Theorem 1, we then apply Lemmas 6 and 7 to Eq. (1) to obtain

$$\mathrm{Adv}^{\mathrm{RoI}}_{\Pi,\mathcal{A},\mathcal{S}}(\mathcal{D}) \leq \varepsilon + 2P^*_{\Pi,pp}(IF \mid q_{\mathrm{ins}}) + (q_{\mathrm{ins}} + 2q_{\mathrm{qry}} + q_{\mathrm{del}}) \cdot P^*_{\Pi,pp}(FP \mid q_{\mathrm{ins}}).$$

Remark 3. We discuss why our results do not directly extend to a setting where $\mathcal{A}$ can access the internal state σ. In Fig. 4, observe that, upon reinsertion of an element not currently in the filter, *Ideal* always samples a fresh $X \leftarrow_\$ \mathfrak{R}$, while G^* inserts the same element again. This choice allowed us to obtain distinguishing bounds involving only the NAI* false positive and insertion failure probabilities. However, this difference is clearly detectable if $\mathcal{A}$ can view σ after reinsertion, leading to *Ideal* and G^* being distinguished with a probability close to 1.

4.2 Guarantees for Counting and Cuckoo filters

In this section, we will use Theorem 1 to give concrete correctness guarantees for Counting and Cuckoo filters.

Corollary 2. *Let $q_{ins}, q_{del}, q_{qry}$ be non-negative integers, and let t_a, $t_d > 0$. Let $F\colon \mathfrak{D} \to \mathfrak{R}$. Let Π be a Counting filter AMQ-PDS with public parameters pp and oracle access to F. If R_K for $K \leftarrow_\$ \mathcal{K}$ is a $(q_{ins} + q_{qry} + q_{del}, t_a + t_d, \varepsilon)$-secure pseudorandom function and $F = R_K$, then Π is $(q_{ins}, q_{qry}, q_{del}, t_a, t_s, t_d, \varepsilon')$-adversarially correct, where $t_s \approx t_a$ and $\varepsilon' = \varepsilon + 2m \cdot \left[\frac{e \cdot q_{ins} \cdot k}{maxVal \cdot m}\right]^{maxVal} + (q_{ins} + 2q_{qry} + q_{del}) \cdot \left[1 - e^{-\frac{(q_{ins}+0.5)k}{m-1}}\right]^k$.*

Proof. From the $\mathsf{ins}, \mathsf{del}, \mathsf{qry}$ algorithms in Fig. 2, we see that Counting filters with oracle access to a random function F are F-decomposable, reinsertion invariant, and satisfy the consistency rules in Def. 12. Further, each $\mathsf{ins}, \mathsf{del}$ and qry call contains one call to the function F. Then, Theorem 1 holds with $\alpha = \beta = \gamma = 1$. Using Lemmas 2 and 3, we obtain the result.

Remark 4. The adversarial correctness bound for Bloom filters in [13, Corollary 4.4] holds for insertion-only Counting filters.

Corollary 3. *Let $q_{ins}, q_{del}, q_{qry}$ be non-negative integers, and let t_a, $t_d > 0$. Let $F\colon \mathfrak{D} \to \mathfrak{R}$. Let Π be a PRF-wrapped Cuckoo filter AMQ-PDS with public parameters pp and oracle access to F. If R_K for $K \leftarrow_\$ \mathcal{K}$ is a $(q_{ins}+q_{qry}+q_{del}, t_a+t_d, \varepsilon)$-secure pseudorandom function and $F = R_K$, then Π is $(q_{ins}, q_{qry}, q_{del}, t_a, t_s, t_d, \varepsilon')$-adversarially correct, where $t_s \approx t_a$ and*

$$\varepsilon' = \varepsilon + \frac{4}{\left(|\mathfrak{R}| \cdot 2^{\lambda_T + \lambda_I - 1}\right)^{s-1}} \binom{q_{ins}}{s} \prod_{i=1}^{s-1} \left[(|\mathfrak{R}| - i)(2^{\lambda_T} - i)\right]$$
$$+ (q_{ins} + 2q_{qry} + q_{del}) \cdot \left[1 - \left(1 - 2^{-\lambda_T}\right)^{2s+1} + \frac{q_{ins}}{|\mathfrak{R}|}\right].$$

Proof. From the $\mathsf{ins}, \mathsf{del}, \mathsf{qry}$ algorithms in Fig. 7, we see that PRF-wrapped Cuckoo filters with oracle access to a random function F are F-decomposable, reinsertion invariant, and satisfy the consistency rules in Def. 12. Further, each $\mathsf{ins}, \mathsf{del}$ and qry call contains one call to the function F. Then, Theorem 1 holds with $\alpha = \beta = \gamma = 1$. Using Lemmas 4 and 5, we obtain the result.

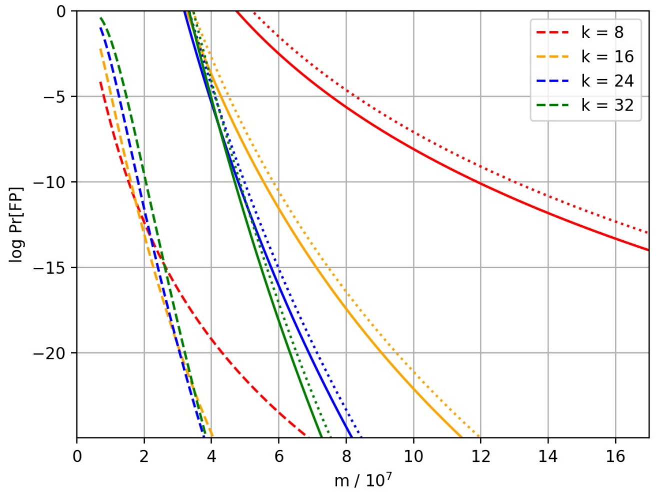

Fig. 6: Correctness guarantees vs. storage trade-offs for Counting filters with $maxVal = 15$ (4 bits counters), $q_{\text{ins}} = q_{\text{del}} = q_{\text{qry}} = 2^{20}$, $\varepsilon = 2^{-128}$. Dashed lines represent non-adversarial guarantees (Lemma 2), solid lines represent adversarial guarantees for the insertion-only setting ([13, Corollary 4.4]), and dotted lines represent adversarial guarantees for the setting with insertions and deletions (Corollary 2).

5 Secure instances

In this section, we outline how our results can be used to secure AMQ-PDS in practice. Let us consider the example outlined in Remark 2, with the predicate $P := [\mathcal{A}\text{'s final } \mathbf{Qry}(x) \text{ query on } x \leftarrow_{\$} \mathfrak{D} \setminus \{x_1, ..., x_n\} \text{ returns } \top]$.

Since Theorem 1 holds for any predicate, the probability of an adversary $\mathcal{A}$ satisfying P in the real world is given by $\Pr[\mathcal{D}(\mathcal{A}){=}1] \leq \varepsilon + 2P^{*}_{\Pi,pp}(IF \mid q_{\text{ins}}) + (q_{\text{ins}} + 2q_{\text{qry}} + q_{\text{del}} + 1) \cdot P^{*}_{\Pi,pp}(FP \mid q_{\text{ins}})$.

We illustrate the behaviour of this bound for the example of Counting filters. In Fig. 6, we plot an upper bound of the false positive probability against the size of the Counting filter in three settings: the non-adversarial setting, the insertion-only adversarial setting, and the setting with deletions studied in this work. By Remark 4, we can analyse the insertion-only setting using the results in [13]: $\Pr[\mathcal{D}(\mathcal{A}){=}1] \leq \varepsilon + (2q_{\text{qry}} + 1) \cdot P_{\Pi,pp}(FP \mid q_{\text{ins}})$.

From Fig. 6, we observe that guaranteeing a specific false positive probability even in an adversarial setting with deletions requires roughly trebling the size of the filter, when compared to the honest (NAI) setting. Crucially, deletions do not incur a significant cost when compared to the insertion-only setting; the additional term of $P^{*}_{\Pi,pp}(IF \mid q_{\text{ins}})$ can be made very small with the choice of an appropriate *maxVal*. For Cuckoo filters, the same observation holds for the choice of λ_I and λ_T. Hence, moving to the more complex scenario of allowing deletions does not hinder the practicality of our results.

Bibliography

[1] Bloom filters and cuckoo filters for cache summarization. https://blog.fleek.network/post/bloom-and-cuckoo-filters-for-cache-summarization/.

[2] Redisbloom: Probabilistic data structures for redis. https://redis.com/modules/redis-bloom/.

[3] Ziv Bar-Yossef, T. S. Jayram, Ravi Kumar, D. Sivakumar, and Luca Trevisan. Counting distinct elements in a data stream. In *International Workshop on Randomization and Approximation Techniques in Computer Science*, 2002. https://doi.org/10.1007/3-540-45726-7_1.

[4] Michael A. Bender, Martin Farach-Colton, Mayank Goswami, Rob Johnson, Samuel McCauley, and Shikha Singh. Bloom filters, adaptivity, and the dictionary problem. In *FOCS*, 2018. https://doi.org/10.1109/FOCS.2018.00026.

[5] Burton H. Bloom. Space/time trade-offs in hash coding with allowable errors. *Communications of the ACM*, 13(7):422–426, 1970. https://doi.org/10.1145/362686.362692.

[6] Andrei Z. Broder and Michael Mitzenmacher. Survey: Network applications of Bloom filters: A survey. *Internet Mathematics*, 1(4):485–509, 2003. https://doi.org/10.1080/15427951.2004.10129096.

[7] Yan-Cheng Chang and Michael Mitzenmacher. Privacy preserving keyword searches on remote encrypted data. In *Applied Cryptography and Network Security*, 2005. https://doi.org/10.1007/11496137_30.

[8] David Clayton, Christopher Patton, and Thomas Shrimpton. Probabilistic data structures in adversarial environments. In *ACM SIGSAC CCS*, 2019. https://doi.org/10.1145/3319535.3354235.

[9] Graham Cormode and S. Muthukrishnan. An improved data stream summary: the count-min sketch and its applications. *Journal of Algorithms*, 55(1):58–75, 2005. https://doi.org/10.1016/j.jalgor.2003.12.001.

[10] Bin Fan, Dave G. Andersen, Michael Kaminsky, and Michael D. Mitzenmacher. Cuckoo filter: Practically better than Bloom. In *CoNEXT*, 2014. https://doi.org/10.1145/2674005.2674994.

[11] Bin Fan, David G. Andersen, and Michael Kaminsky. Cuckoo filter reference implementation. https://github.com/efficient/cuckoofilter/blob/917583d6abef692dfa8e14453bd77d6e0b61eef3/src/cuckoofilter.h#L139, 2013.

[12] Li Fan, Pei Cao, J. Almeida, and A.Z. Broder. Summary cache: a scalable wide-area web cache sharing protocol. *IEEE/ACM Transactions on Networking*, 8(3):281–293, 2000. https://doi.org/10.1109/90.851975.

[13] Mia Filić, Kenny Paterson, Anupama Unnikrishnan, and Fernando Virdia. Adversarial correctness and privacy for probabilistic data structures. In *ACM SIGSAC CCS*, 2022. https://doi.org/10.1145/3548606.3560621.

[14] Philippe Flajolet, Éric Fusy, Olivier Gandouet, and Frédéric Meunier. Hyperloglog: the analysis of a near-optimal cardinality estimation algorithm. In *Conference on Analysis of Algorithms*, 2007. https://doi.org/10.46298/dmtcs.3545.

[15] Sergio Galán, Pedro Reviriego, Stefan Walzer, Alfonso Sánchez-Macian, Shanshan Liu, and Fabrizio Lombardi. On the privacy of counting bloom filters under a black-box attacker. *IEEE Transactions on Dependable and Secure Computing*, 20(5), 2023. https://doi.org/10.1109/TDSC.2022.3217115.

[16] Thomas Gerbet, Amrit Kumar, and Cédric Lauradoux. The power of evil choices in Bloom filters. In *IEEE/IFIP Conference on Dependable Systems and Networks*, 2015. https://doi.org/10.1109/DSN.2015.21.

[17] Junzhi Gong, Tong Yang, Haowei Zhang, Hao Li, Steve Uhlig, Shigang Chen, Lorna Uden, and Xiaoming Li. HeavyKeeper: An accurate algorithm for finding top-k elephant flows. In *USENIX Annual Technical Conference*, 2018. https://doi.org/10.1109/TNET.2019.2933868.

[18] Laura Hetz, Thomas Schneider, and Christian Weinert. Scaling mobile private contact discovery to billions of users. In *ESORICS*, 2023. https://doi.org/10.1007/978-3-031-50594-2_23.

[19] Stefan Heule, Marc Nunkesser, and Alexander Hall. Hyperloglog in practice: Algorithmic engineering of a state of the art cardinality estimation algorithm. In *Conference on Extending Database Technology*, 2013. https://doi.org/10.1145/2452376.2452456.

[20] Daniel Kales, Christian Rechberger, Thomas Schneider, Matthias Senker, and Christian Weinert. Mobile private contact discovery at scale. In *USENIX Security*, 2019.

[21] Tsvi Kopelowitz, Samuel McCauley, and Ely Porat. Support optimality and adaptive cuckoo filters. In *Algorithms and Data Structures*, 2021. https://doi.org/10.1007/978-3-030-83508-8_40.

[22] Anukool Lakhina, Mark Crovella, and Christiphe Diot. Characterization of network-wide anomalies in traffic flows. In *ACM SIGCOMM Conference on Internet Measurement*, 2004. https://doi.org/10.1145/1028788.1028813.

[23] James Larisch, David Choffnes, Dave Levin, Bruce M Maggs, Alan Mislove, and Christo Wilson. Crlite: A scalable system for pushing all tls revocations to all browsers. In *IEEE S&P*, 2017. https://doi.org/10.1109/SP.2017.17.

[24] Yehuda Lindell. How to simulate it – a tutorial on the simulation proof technique, 2017. https://doi.org/10.1007/978-3-319-57048-8_6.

[25] Linsheng Liu, Daniel S. Roche, Austin Theriault, and Arkady Yerukhimovich. Fighting fake news in encrypted messaging with the fuzzy anonymous complaint tally system (facts). In *Network and Distributed Systems Security Symposium*, 2022. https://doi.org/10.14722/ndss.2022.23109.

[26] Lailong Luo, Deke Guo, Richard T. B. Ma, Ori Rottenstreich, and Xueshan Luo. Optimizing bloom filter: Challenges, solutions, and comparisons. *IEEE*

Communications Surveys & Tutorials, 21(2):1912–1949, 2019. https://doi.org/10.1109/COMST.2018.2889329.

[27] Sam A. Markelon, Mia Filić, and Thomas Shrimpton. Compact frequency estimators in adversarial environments. In *ACM SIGSAC CCS*, 2023. https://doi.org/10.1145/3576915.3623216.

[28] Luca Melis, George Danezis, and Emiliano De Cristofaro. Efficient private statistics with succinct sketches. In *Network and Distributed Systems Security Symposium*, 2016. https://doi.org/10.14722/ndss.2016.23175.

[29] Páll Melsted and Jonathan K Pritchard. Efficient counting of k-mers in dna sequences using a bloom filter. *BMC Bioinformatics*, 12, 2011. https://doi.org/10.1186/1471-2105-12-333.

[30] Moni Naor and Noa Oved. Bet-or-pass: Adversarially robust bloom filters. In *TCC*, 2022. https://doi.org/10.1007/978-3-031-22365-5_27.

[31] Moni Naor and Eylon Yogev. Bloom filters in adversarial environments. In *CRYPTO*, 2015. https://doi.org/10.1007/978-3-662-48000-7_28.

[32] Moni Naor and Eylon Yogev. Bloom filters in adversarial environments. *ACM Transactions on Algorithms*, 15(3):35:1–35:30, 2019. https://doi.org/10.1145/3306193.

[33] Kenneth G. Paterson and Mathilde Raynal. HyperLogLog: Exponentially bad in adversarial settings. In *EuroS&P*, 2022. https://doi.org/10.1109/EuroSP53844.2022.00018.

[34] Henning Perl, Yassene Mohammed, Michael Brenner, and Matthew Smith. Fast confidential search for bio-medical data using bloom filters and homomorphic cryptography. *IEEE Conference on E-Science*, pages 1–8, 2012. https://doi.org/10.1109/eScience.2012.6404484.

[35] Xiaofeng Shi, Shouqian Shi, Minmei Wang, Jonne Kaunisto, and Chen Qian. On-device iot certificate revocation checking with small memory and low latency. In *ACM SIGSAC CCS*, 2021. https://doi.org/10.1145/3460120.3484580.

[36] Dimitrios Sikeridis, Sean Huntley, David Ott, and Michael Devetsikiotis. Intermediate certificate suppression in post-quantum tls: An approximate membership querying approach. In *CoNEXT*, 2022. https://doi.org/10.1145/3555050.3569127.

[37] Henrik Stranneheim, Max Käller, Tobias Allander, Björn Andersson, Lars Arvestad, and Joakim Lundeberg. Classification of dna sequences using bloom filters. volume 26, pages 1595–1600, 2010. https://doi.org/10.1093/bioinformatics/btq230.

[38] Jeff Yan and Pook Leong Cho. Enhancing collaborative spam detection with bloom filters. In *Annual Computer Security Applications Conference*, 2006. https://doi.org/10.1109/ACSAC.2006.26.

[39] Kevin Yeo. Cuckoo hashing in cryptography: Optimal parameters, robustness and applications. In *CRYPTO*, 2023. https://doi.org/10.1007/978-3-031-38551-3_7.

A Counting Filters

In Lemma 2, we compute the maximal NAI* false positive probability for Counting filters.

Lemma 2. *Let Π be an $(m, k, maxVal)$-Counting filter using a random function $F : \mathfrak{D} \to [m]^k$. Then, for any n, $P^*_{\Pi,pp}(FP \,|\, n) \leq \left[1 - e^{-\frac{(n+0.5)k}{m-1}}\right]^k$.*

Proof. Recall the maximal NAI* false positive probability definition (Def. 8), where an algorithm $\mathcal{U}^{\mathrm{Id}_{\mathfrak{R}}}_{\Pi,pp}$ constructs an NAI* state for a Counting filter with public parameters $pp = (m, k, maxVal)$ with the maximal false positive probability using $X_1, ..., X_n$. To calculate the maximal false positive probability for Counting filters, we would need to construct this algorithm $\mathcal{U}^{\mathrm{Id}_{\mathfrak{R}}}_{\Pi,pp}$.

We start by considering a $\mathcal{U}^{\mathrm{Id}_{\mathfrak{R}}}_{\Pi,pp}$ that inserts all $X_1, ..., X_n$, since the state with the maximal false positive probability maximises the number of non-zero counters, and the false positive probability cannot decrease with each insertion. However, such a $\mathcal{U}^{\mathrm{Id}_{\mathfrak{R}}}_{\Pi,pp}$ does not necessarily lead to the maximal false positive probability due to the possibility of failed insertions. For example, consider two elements, X_i and X_j, for $i < j < n$, which share a counter. If that counter reaches $maxVal$ with the insertion of X_i, any later insertion of X_j will fail. However, it may be that the insertion of X_j would lead to more non-zero counters in σ than the insertion of X_i, and so the false positive probability may be higher by not inserting X_i at all. Therefore, the construction of $\mathcal{U}^{\mathrm{Id}_{\mathfrak{R}}}_{\Pi,pp}$ for $maxVal < \infty$ (and thus an exact calculation of $P^*_{\Pi,pp}(FP \,|\, n)$) is not so straightforward. However, we will show that it is still possible to upper bound $P^*_{\Pi,pp}(FP \,|\, n)$ using the existing results on false positive probability from Lemma 1.

To do so, let us consider a Counting filter with public parameters $pp' = (m, k, \infty)$. Since every insertion will succeed when $maxVal$ is equal to ∞, we can define $\mathcal{U}^{\mathrm{Id}_{\mathfrak{R}}}_{\Pi,pp'}$ as an algorithm that simply inserts all $X_1, ..., X_n$.

Let σ be the output of $\mathcal{U}^{\mathrm{Id}_{\mathfrak{R}}}_{\Pi,pp}$, and let Δ be the output of $\mathcal{U}^{\mathrm{Id}_{\mathfrak{R}}}_{\Pi,pp'}$. Regardless of the definition of $\mathcal{U}^{\mathrm{Id}_{\mathfrak{R}}}_{\Pi,pp}$, running $\mathcal{U}^{\mathrm{Id}_{\mathfrak{R}}}_{\Pi,pp}$ and $\mathcal{U}^{\mathrm{Id}_{\mathfrak{R}}}_{\Pi,pp'}$ on the same input will result in at least all counters that are non-zero in σ being non-zero in Δ. Therefore,

$$\begin{aligned}
P^*_{\Pi,pp}(FP \,|\, n) &\leq P^*_{\Pi,pp'}(FP \,|\, n) \\
&= \Pr\left[\begin{array}{c} X_1, ..., X_n \leftarrow\!\$\; \mathfrak{R} \\ \Delta \leftarrow \mathcal{B}^{\mathrm{Id}_{\mathfrak{R}}}_{\Pi,pp'}(X_1, ..., X_n) : \\ \top \leftarrow\!\$\; \mathsf{qry}^{\mathrm{Id}_{\mathfrak{R}}}(X \leftarrow\!\$\; \mathfrak{R}, \Delta) \end{array}\right] \\
&= \Pr\left[\begin{array}{c} X_1, ..., X_n \leftarrow\!\$\; \mathfrak{R} \\ \Delta \leftarrow \mathsf{setup}(m, k, \infty) \\ \text{for } i \in [n] : \quad (\top, \Delta) \leftarrow\!\$\; \mathsf{ins}^{\mathrm{Id}_{\mathfrak{R}}}(X_i, \Delta) : \\ \top \leftarrow\!\$\; \mathsf{qry}^{\mathrm{Id}_{\mathfrak{R}}}(X \leftarrow\!\$\; \mathfrak{R}, \Delta) \end{array}\right] \\
&= P_{\Pi,pp}(FP \,|\, n),
\end{aligned}$$

where the last line follows from Lemma 1.

In Lemma 3, we compute the maximal NAI* insertion failure probability for Counting filters.

Lemma 3. *Let Π be an $(m, k, maxVal)$-Counting filter using a random function $F : \mathfrak{D} \to [m]^k$. Then, for any n, $P^*_{\Pi,pp}(IF \mid n) \leq m \cdot \left[\frac{e \cdot n \cdot k}{maxVal \cdot m}\right]^{maxVal}$.*

Proof. Recall the maximal NAI* insertion failure probability definition (Def. 9), where an algorithm $\mathcal{V}^{\mathrm{Id}_{\mathfrak{R}}}_{\Pi,pp}$ constructs an NAI* state for a Counting filter with public parameters $pp = (m, k, maxVal)$ with the maximal probability of an insertion on one of $X_1, \ldots, X_n$ failing. To calculate this probability for Counting filters, we would need to construct this algorithm $\mathcal{V}^{\mathrm{Id}_{\mathfrak{R}}}_{\Pi,pp}$.

First, note that if the insertion of X_l fails, then X_l must be associated with a counter $j \in [m]$ such that $\sigma[j] \geq maxVal$. We can then upper bound this by

$$P^*_{\Pi,pp}(IF \mid n) \leq \Pr\left[\begin{array}{c} X_1, \ldots, X_n \leftarrow\$ \mathfrak{R} \\ \sigma \leftarrow \mathcal{V}^{\mathrm{Id}_{\mathfrak{R}}}_{\Pi,pp}(X_1, \ldots, X_n) : \\ \exists j \in [m] \text{ such that } \sigma[j] \geq maxVal \end{array}\right]$$
$$\leq \Pr\left[\begin{array}{c} limit \leftarrow maxVal \\ X_1, \ldots, X_n \leftarrow\$ \mathfrak{R} \\ \sigma \leftarrow \mathcal{W}^{\mathrm{Id}_{\mathfrak{R}}}_{\Pi,pp}(X_1, \ldots, X_n, limit) : \\ \exists j \in [m] \text{ such that } \sigma[j] \geq limit \end{array}\right],$$

where $\mathcal{W}^{\mathrm{Id}_{\mathfrak{R}}}_{\Pi,pp}(X_1,...,X_n, limit)$ outputs an NAI* state using $\mathsf{ins}^{\mathrm{Id}_{\mathfrak{R}}}(\cdot,\sigma), \mathsf{del}^{\mathrm{Id}_{\mathfrak{R}}}(\cdot,\sigma)$ on $X_1, ..., X_n$ that has the maximal probability of one of its counters being greater than or equal to *limit*.

Although the latter probability increases with n, in general, we cannot construct $\mathcal{W}^{\mathrm{Id}_{\mathfrak{R}}}_{\Pi,pp}$ by simply inserting all $X_1, \ldots, X_n$ due to insertion failures (as in Lemma 2). Therefore, we will again upper bound this probability instead of directly calculating it.

Let $\overline{\mathsf{ins}}$ denote a modified version of the ins algorithm in Fig. 2, where the counters are *always* incremented upon insertion, i.e. the check in line 6 is skipped. Let $\overline{\mathcal{W}}^{\mathrm{Id}_{\mathfrak{R}}}_{\Pi,pp}$ be the same as $\mathcal{W}^{\mathrm{Id}_{\mathfrak{R}}}_{\Pi,pp}$ but where insertions are done using $\overline{\mathsf{ins}}$. Consider a Counting filter with public parameters $pp' = (m, k, \infty)$, such that no insertion can fail. Then, we can define $\overline{\mathcal{W}}^{\mathrm{Id}_{\mathfrak{R}}}_{\Pi,pp'}$ as an algorithm that simply inserts all $X_1, \ldots, X_n$.

Let σ be the output of $\mathcal{W}^{\mathrm{Id}_{\mathfrak{R}}}_{\Pi,pp}$, and let Δ be the output of $\overline{\mathcal{W}}^{\mathrm{Id}_{\mathfrak{R}}}_{\Pi,pp'}$. Running $\mathcal{W}^{\mathrm{Id}_{\mathfrak{R}}}_{\Pi,pp}$ and $\overline{\mathcal{W}}^{\mathrm{Id}_{\mathfrak{R}}}_{\Pi,pp'}$ on the same input will result in $\sigma[j] \leq \Delta[j]$ for every counter $j \in [m]$, giving

$$P^*_{\Pi,pp}(IF \mid n) \leq \Pr\left[\begin{array}{c} X_1, \ldots, X_n \leftarrow\$ \mathfrak{R} \\ \Delta \leftarrow \mathsf{setup}(m, k, \infty) \\ \text{for } i \in [n] : \ (\top, \Delta) \leftarrow \overline{\mathsf{ins}}^{\mathrm{Id}_{\mathfrak{R}}}(X_i, \Delta) : \\ \exists j \in [m] \text{ such that } \Delta[j] \geq maxVal \end{array}\right]$$
$$\leq m \cdot \left[\frac{e \cdot n \cdot k}{maxVal \cdot m}\right]^{maxVal},$$

where the last line follows from [12].

B Cuckoo Filters

In Fig. 7, we give the AMQ-PDS syntax instantiation for PRF-wrapped Cuckoo filters.

In Lemma 4, we compute the maximal NAI* false positive probability for PRF-wrapped Cuckoo filters.

Lemma 4. *Let Π be an $(s, \lambda_I, \lambda_T, num)$-PRF-wrapped Cuckoo filter using random functions $F : \mathfrak{D} \to \mathfrak{R}$, $H_T : \mathfrak{D} \to \{0,1\}^{\lambda_T}$, and $H_I : \mathfrak{D} \to \{0,1\}^{\lambda_I}$. Then, for any n, $P^*_{\Pi,pp}(FP \mid n) \leq 1 - \left(1 - 2^{-\lambda_T}\right)^{2s+1} + \frac{n}{|\mathfrak{R}|}$.*

Proof. In [10], an upper bound was proven on the false positive probability when querying a randomly sampled element on *any* state, i.e. not necessarily an NAI* state. To derive this bound, it is assumed that the "worst" state $\sigma_{\max}$ can be constructed, which contains a distinct fingerprint in every slot, and that the queried element is not among elements $V_{\max}$ used to create the state.

Thus, from Def. 8, the result follows from [10], but accounting for a random element from the range of F equalling one of the n elements used to create the state. Thus,

$$\begin{aligned}
P^*_{\Pi,pp}(FP \mid n) &:= \Pr\begin{bmatrix} \sigma \leftarrow \mathsf{setup}(s, \lambda_I, \lambda_T, num) \\ X_1, \ldots, X_n \leftarrow_{\$} \mathfrak{R} \\ V \leftarrow X_1, \ldots, X_n \\ \sigma \leftarrow \mathcal{U}^{\mathrm{Id}_{\mathfrak{R}}}_{\Pi,pp}(X_1, \ldots, X_n) : \\ \top \leftarrow_{\$} \mathsf{qry}^{\mathrm{Id}_{\mathfrak{R}}}(X \leftarrow_{\$} \mathfrak{R}, \sigma) \end{bmatrix} \\
&\leq \Pr\begin{bmatrix} \sigma \leftarrow \mathsf{setup}(s, \lambda_I, \lambda_T, num) \\ X_1, \ldots, X_n \leftarrow_{\$} \mathfrak{R} \\ V \leftarrow X_1, \ldots, X_n \\ \sigma \leftarrow \mathcal{U}^{\mathrm{Id}_{\mathfrak{R}}}_{\Pi,pp}(X_1, \ldots, X_n) : \\ \top \leftarrow_{\$} \mathsf{qry}^{\mathrm{Id}_{\mathfrak{R}}}(X \leftarrow_{\$} \mathfrak{R} \backslash V, \sigma) \end{bmatrix} + \Pr\begin{bmatrix} \sigma \leftarrow \mathsf{setup}(s, \lambda_I, \lambda_T, num) \\ X_1, \ldots, X_n \leftarrow_{\$} \mathfrak{R} \\ V \leftarrow X_1, \ldots, X_n \\ \sigma \leftarrow \mathcal{U}^{\mathrm{Id}_{\mathfrak{R}}}_{\Pi,pp}(X_1, \ldots, X_n) : \\ [X \leftarrow_{\$} \mathfrak{R} \in V] \end{bmatrix} \\
&\leq \Pr\left[\top \leftarrow_{\$} \mathsf{qry}^{\mathrm{Id}_{\mathfrak{R}}}(X \leftarrow_{\$} \mathfrak{R} \backslash V_{\max}, \sigma_{\max})\right] + \Pr\begin{bmatrix} X_1, \ldots, X_n \leftarrow_{\$} \mathfrak{R} : \\ [X \leftarrow_{\$} \mathfrak{R} \in V] \end{bmatrix} \\
&\leq \Pr\left[\top \leftarrow_{\$} \mathsf{qry}^{\mathrm{Id}_{\mathfrak{R}}}(X \leftarrow_{\$} \mathfrak{R} \backslash V_{\max}, \sigma_{\max})\right] + \frac{n}{|\mathfrak{R}|}, \\
&\leq 1 - \left(1 - 2^{-\lambda_T}\right)^{2s+1} + \frac{n}{|\mathfrak{R}|},
\end{aligned}$$

where the last step follows from [10].

In Lemma 5, we compute the maximal NAI* insertion failure probability for PRF-wrapped Cuckoo filters.

Lemma 5. *Let Π be an $(s, \lambda_I, \lambda_T, num)$-PRF-wrapped Cuckoo filter using random functions $F : \mathfrak{D} \to \mathfrak{R}$, $H_T : \mathfrak{D} \to \{0,1\}^{\lambda_T}$, and $H_I : \mathfrak{D} \to \{0,1\}^{\lambda_I}$. Then, for any n,*

$$P^*_{\Pi,pp}(IF \mid n) \leq \frac{2}{\left(|\mathfrak{R}| \cdot 2^{\lambda_T + \lambda_I - 1}\right)^{s-1}} \binom{n}{s} \prod_{i=1}^{s-1} \left[(|\mathfrak{R}| - i)(2^{\lambda_T} - i)\right].$$

Proof. Recall the maximal NAI* insertion failure probability definition (Def. 9). If the insertion of X_l in σ fails, then X_l must be associated with two distinct buckets $j_1, j_2 \in [2^{\lambda_I}]$ such that $\text{load}(\sigma[j_1]) = \text{load}(\sigma[j_2]) = s$. Therefore, we can upper bound this by

$$\begin{aligned}
P^*_{\Pi,pp}(IF \mid n) &\leq \Pr\left[\begin{array}{c} X_1, ..., X_n \leftarrow\!\$\, \mathfrak{R} \\ \sigma \leftarrow \mathcal{V}^{\mathrm{Id}_{\mathfrak{R}}}_{\Pi,pp}(X_1, ..., X_n) : \\ \exists j_1, j_2 \in [2^{\lambda_I}] \text{ such that} \\ [\text{load}(\sigma[j_1]) = s] \wedge [\text{load}(\sigma[j_2]) = s] \end{array}\right] \\
&\leq \Pr\left[\begin{array}{c} X_1, ..., X_n \leftarrow\!\$\, \mathfrak{R} \\ \sigma \leftarrow \mathcal{V}^{\mathrm{Id}_{\mathfrak{R}}}_{\Pi,pp}(X_1, ..., X_n) : \\ \exists j \in [2^{\lambda_I}] \text{ such that } \text{load}(\sigma[j]) \geq s \end{array}\right] \\
&\leq \Pr\left[\begin{array}{c} \mathit{limit} \leftarrow s \\ X_1, ..., X_n \leftarrow\!\$\, \mathfrak{R} \\ \sigma \leftarrow \mathcal{W}^{\mathrm{Id}_{\mathfrak{R}}}_{\Pi,pp}(X_1, ..., X_n, \mathit{limit}) : \\ \exists j \in [2^{\lambda_I}] \text{ such that } \text{load}(\sigma[j]) \geq \mathit{limit} \end{array}\right],
\end{aligned}$$

where $\mathcal{W}^{\mathrm{Id}_{\mathfrak{R}}}_{\Pi,pp}(X_1,...,X_n, \mathit{limit})$ outputs an NAI* state using $\mathsf{ins}^{\mathrm{Id}_{\mathfrak{R}}}(\cdot,\sigma), \mathsf{del}^{\mathrm{Id}_{\mathfrak{R}}}(\cdot,\sigma)$ on $X_1,...,X_n$ that has the maximal probability of one of its buckets having at least *limit* fingerprints.

Let $\overline{\mathsf{ins}}$ denote a modified version of the ins algorithm in Fig. 7, where an element's tag is added to *both* of its buckets, if they do not already contain it. Let $\overline{\mathcal{W}}^{\mathrm{Id}_{\mathfrak{R}}}_{\Pi,pp}$ be the same as $\mathcal{W}^{\mathrm{Id}_{\mathfrak{R}}}_{\Pi,pp}$ but where insertions are done using $\overline{\mathsf{ins}}$. Consider a Cuckoo filter with public parameters $pp' = (\infty, \lambda_I, \lambda_T, \mathit{num})$, such that no insertion can fail. Then, we can define $\overline{\mathcal{W}}^{\mathrm{Id}_{\mathfrak{R}}}_{\Pi,pp'}$ as an algorithm that simply inserts all $X_1, ..., X_n$.

Let σ be the output of $\mathcal{W}^{\mathrm{Id}_{\mathfrak{R}}}_{\Pi,pp}$, and let Δ be the output of $\overline{\mathcal{W}}^{\mathrm{Id}_{\mathfrak{R}}}_{\Pi,pp'}$. Running $\mathcal{W}^{\mathrm{Id}_{\mathfrak{R}}}_{\Pi,pp}$ and $\overline{\mathcal{W}}^{\mathrm{Id}_{\mathfrak{R}}}_{\Pi,pp'}$ on the same input will result in the set of tags in bucket $\sigma[j]$ being a subset of tags stored in bucket $\Delta[j]$ for every bucket $j \in [2^{\lambda_I}]$, giving

$$\begin{aligned}
P^*_{\Pi,pp}(IF \mid n) &\leq \Pr\left[\begin{array}{c} \mathit{limit} \leftarrow s \\ X_1, ..., X_n \leftarrow\!\$\, \mathfrak{R} \\ \sigma \leftarrow \mathcal{W}^{\mathrm{Id}_{\mathfrak{R}}}_{\Pi,pp}(X_1, ..., X_n, \mathit{limit}) : \\ \exists j \in [2^{\lambda_I}] \text{ such that } \text{load}(\sigma[j]) \geq \mathit{limit} \end{array}\right] \\
&\leq \Pr\left[\begin{array}{c} \mathit{limit} \leftarrow s \\ X_1, ..., X_n \leftarrow\!\$\, \mathfrak{R} \\ \Delta \leftarrow \overline{\mathcal{W}}^{\mathrm{Id}_{\mathfrak{R}}}_{\Pi,pp'}(X_1, ..., X_n, \mathit{limit}) : \\ \exists j \in [2^{\lambda_I}] \text{ such that } \text{load}(\Delta[j]) \geq \mathit{limit} \end{array}\right] \\
&= \Pr\left[\begin{array}{c} \Delta \leftarrow \mathsf{setup}(\infty, \lambda_I, \lambda_T, \mathit{num}) \\ X_1, ..., X_n \leftarrow\!\$\, \mathfrak{R} \\ \text{for } i \in [n] : \ (\top, \Delta) \leftarrow \overline{\mathsf{ins}}^{\mathrm{Id}_{\mathfrak{R}}}(X_i, \Delta) : \\ \exists j \in [2^{\lambda_I}] \text{ such that } \text{load}(\Delta[j]) \geq s \end{array}\right].
\end{aligned}$$

There are 2^{λ_I} possible choices of j, and $\binom{n}{s}$ possible choices of s elements whose first or second bucket is j. Now, the s elements must be distinct in order to be added to their buckets; since we are considering PRF-wrapped Cuckoo filters, this means they must be distinct even after they are passed through $F : \mathfrak{D} \to \mathfrak{R}$, which occurs with probability $[1 \cdot \left(\frac{|\mathfrak{R}|-1}{|\mathfrak{R}|}\right) \cdot ... \left(\frac{|\mathfrak{R}|-(s-1)}{|\mathfrak{R}|}\right)]$. Then, observe that we also require the tags of the s elements to be distinct, so that they are added to their buckets. This occurs with probability $[1 \cdot \left(\frac{2^{\lambda_T}-1}{2^{\lambda_T}}\right) \cdot ... \left(\frac{2^{\lambda_T}-(s-1)}{2^{\lambda_T}}\right)]$. Finally, we compute the probability that the s distinct elements have their first or second bucket equal to j, which is $\left(\frac{2}{2^{\lambda_I}}\right)^s$. Putting this together, we obtain

$$\begin{aligned} P^*_{\Pi,pp}(IF \mid n) &\leq 2^{\lambda_I} \binom{n}{s} \Big[\frac{(|\mathfrak{R}|-1) \cdot ...(|\mathfrak{R}|-(s-1))}{|\mathfrak{R}|^{s-1}} \Big] \\ &\qquad \times \Big[\frac{(2^{\lambda_T}-1) \cdot ...(2^{\lambda_T}-(s-1))}{2^{\lambda_T \cdot (s-1)}} \Big] \left(\frac{2}{2^{\lambda_I}} \right)^s \\ &= \frac{2}{\left(|\mathfrak{R}| \cdot 2^{\lambda_T+\lambda_I-1}\right)^{s-1}} \binom{n}{s} \prod_{i=1}^{s-1} \big[(|\mathfrak{R}|-i)(2^{\lambda_T}-i) \big]. \end{aligned}$$

```
setup(pp)
$s, \lambda_I, \lambda_T, num \leftarrow pp$
// initialise $2^{\lambda_I}$ buckets
// and $s$ $\lambda_T$-bit slots
for $i \in [2^{\lambda_I}]$
    $\sigma_i \leftarrow \bot^s$
// stashed element bucket
$k_{stash} \leftarrow \bot$
// stashed element tag
$\sigma_{stash} \leftarrow \bot$
$\sigma \leftarrow (\sigma_i)_{i \in [2^{\lambda_I}]}, \sigma_{stash}, k_{stash}$
return $\sigma$
```

```
qry$^{F,H_T,H_I}(x, \sigma)$
$x \leftarrow F(x)$
$tag \leftarrow H_T(x)$
$i_1 \leftarrow H_I(x)$
$i_2 \leftarrow i_1 \oplus H_I(tag)$
// tag in stash?
$a \leftarrow (tag = \sigma_{stash})$
$a \leftarrow a \wedge (k_{stash} \in \{i_1, i_2\})$
// tag in bucket?
$a \leftarrow a \vee tag \in \sigma_{i_1} \vee tag \in \sigma_{i_2}$
return $a$
```

```
ins$^{F,H_T,H_I}(x, \sigma)$
$x \leftarrow F(x)$
$tag \leftarrow H_T(x)$
$i_1 \leftarrow H_I(x)$
$i_2 \leftarrow i_1 \oplus H_I(tag)$
$a \leftarrow (tag = \sigma_{stash})$
$a \leftarrow a \wedge (k_{stash} \in \{i_1, i_2\})$
$a \leftarrow a \vee tag \in \sigma_{i_1} \vee tag \in \sigma_{i_2}$
if $a = \top$ : return $\top, \sigma$
// ins disabled?
if $\sigma_{stash} \neq \bot$ : return $\bot, \sigma$
// check if empty slots
for $i \in \{i_1, i_2\}$
    if load$(\sigma_i) < s$
        $\sigma_i \leftarrow \sigma_i \diamond tag$
        return $\top, \sigma$
// displace something
$i \leftarrow\$ \{i_1, i_2\}$
$\sigma \leftarrow evict^{H_I}(i, tag, \sigma)$
return $\top, \sigma$
```

```
$evict^{H_I}(i, tag, \sigma)$
for $g \in [num]$
    $slot \leftarrow\$ [s]$
    $elem \leftarrow \sigma_i[slot]$
    // swap elem, tag
    $\sigma_i[slot] \leftarrow tag;\ tag \leftarrow elem$
    $i \leftarrow i \oplus H_I(tag)$
    if load$(\sigma_i) < s$
        $\sigma_i \leftarrow \sigma_i \diamond tag$
        return $\top, \sigma$
$k_{stash} \leftarrow i$
$\sigma_{stash} \leftarrow tag$
return $\sigma$
```

```
del$^{F,H_T,H_I}(x, \sigma)$
$x \leftarrow F(x)$
$tag \leftarrow H_T(x)$
$i_1 \leftarrow H_I(x)$
$i_2 \leftarrow i_1 \oplus H_I(tag)$
// tag in stash?
if $[tag = \sigma_{stash}]$
    if $(k_{stash} \in \{i_1, i_2\})$
        $\sigma_{stash} \leftarrow \bot$
        $k_{stash} \leftarrow \bot$
        return $\top, \sigma$
// tag in bucket?
for $i \in \{i_1, i_2\}$
    if $tag \in \sigma_i$
        $\sigma_i \leftarrow \sigma_i \setminus tag$
        // empty the stash
        if $\sigma_{stash} \neq \bot$
            $j_1 \leftarrow k_{stash}$
            $j_2 \leftarrow j_1 \oplus H_I(\sigma_{stash})$
            for $j \in \{j_1, j_2\}$
                if load$(\sigma_j) < s$
                    $\sigma_j \leftarrow \sigma_j \diamond \sigma_{stash}$
                    $\sigma_{stash} \leftarrow \bot$
                    $k_{stash} \leftarrow \bot$
        // displace something
        if $\sigma_{stash} \neq \bot$
            $j \leftarrow\$ \{j_1, j_2\}$
            $\sigma \leftarrow evict^{H_I}(j, \sigma_{stash}, \sigma)$
        return $\top, \sigma$
// nothing to delete
return $\bot, \sigma$
```

Fig. 7: AMQ-PDS syntax instantiation for the PRF-wrapped Cuckoo filter. Following the reference implementation [11] by the authors of [10], after we delete an element, we try to empty the stash by re-inserting the stashed element. We write the procedure *evict* separately for ease of understanding.

C More on comparison with [8]

Clayton *et al.* [8] focused on the adversarial correctness of Bloom and Counting filters, as concrete instantiations of their general PDS syntax (which is the basis of ours). They employed a game-based formalism, which required defining a specific winning condition for the adversary, i.e. finding a certain number of false positives or false negatives. In contrast, we use a simulation-based approach to analyse correctness. The power of the simulation-based approach in analysing correctness is that it covers all adversarial goals. In practice, this means that one only needs to compute the probability of achieving a particular goal in the honest setting (which is well-studied in the PDS literature) to upper bound the probability of achieving it in the adversarial setting.

We analyse a class of AMQ-PDS with deletions, focusing on Counting and Cuckoo filters with deletions. Counting filters generalise Bloom filters to allow for deletions. With 1-bit counters and no deletions allowed, Counting filters effectively become Bloom filters. However, Bloom filters do not stop accepting insertions when some of their bit positions are set to their maximum value of 1. Therefore, it is straightforward to see that setting the number of deletions and the insertion failure probability to 0 in our Counting filter results yields bounds for Bloom filters.

While [8] also analysed Counting filters, they examine an altered "ℓ - thresholded" variant, a variant of Counting filters where insertions are disallowed if more than ℓ counters are set. Clayton *et al.* [8] proved the variant to be adversarially correct against a goal of finding r false positives or negatives. Their proof's complexity stems from the proof's reliance on the chosen PDS and adversarial goal. However, it is also important to study the difficulty of making an insertion to the PDS fail in adversarial settings, as this tells us about the difficulty of denial-of-service-style attacks against the structure. Unfortunately, [8] did not analyse this, and their results are not easily transferable to address alternative adversarial goals, such as insertion failure. Moreover, extending [8] to a slight variant of any of their focus PDS, like traditional Counting filters, requires a new proof of similar complexity, as evidenced by their separate analysis of ℓ-thresholded and traditional Bloom filters.

With our approach, (a) the process of deriving results for additional adversarial goals for the same PDS is simplified, and (b) the effort required to transform results for a PDS to its variant is reduced by focusing on NAI* probabilities, which inherently exhibit closer relations among the same events for slight PDS variations compared to those in adversarial settings.

Clayton *et al.* [8] did not analyse Cuckoo filters. However, they analysed Count-Min sketches (CMS), but again a version with ℓ-thresholding. As CMS are primarily designed for frequency rather than membership queries, they fall outside the scope of our paper due to their primary functionality goal. The CMS analysis in [8] is worthwhile but has two limitations: (a) it examines an impractical version due to ℓ-thresholding; (b) their adversarial objective does not align with typical ways of compromising CMS correctness, which considers the magnitude of overestimates, and not their count. A more suitable adversarial goal for the CMS was discussed in [27].

D Recovering results for insertion-only AMQ-PDS from [13]

In the following, we derive results for the class of insertion-only AMQ-PDS analysed in [13]. In general, this class is characterised by the following consistency rules.

Definition 15 (Insertion-only AMQ-PDS consistency rules [13]). *Consider an AMQ-PDS Π. We say Π has:*

– Element permanence *if for all $x \in \mathfrak{D}$, $\sigma \in \Sigma$ such that $\top \leftarrow \mathsf{qry}(x, \sigma)$, and for any sequence of insertions resulting in a later state σ', $b \leftarrow \mathsf{qry}(x, \sigma') \implies b = \top$.*
– Permanent disabling *if given $\sigma \in \Sigma$ such that there exists $x \in \mathfrak{D}$, $r \in \mathcal{R}$ where $(b, \overline{\sigma}) \leftarrow \mathsf{up}(x, \sigma; r)$ and $b = \bot$, then $\overline{\sigma} = \sigma$ and for any $x' \in \mathfrak{D}$, $r' \in \mathcal{R}$, $(b', \sigma') \leftarrow \mathsf{up}(x', \sigma; r') \Rightarrow b' = \bot$ and $\sigma' = \sigma$.*
– Non-decreasing membership probability *if for all $\sigma \in \Sigma$, $x, y \in \mathfrak{D}$, $r \in \mathcal{R}$, $(b, \sigma') \leftarrow \mathsf{up}(x, \sigma; r) \Rightarrow \Pr[\top \leftarrow \mathsf{qry}(y, \sigma)] \leq \Pr[\top \leftarrow \mathsf{qry}(y, \sigma')]$.*
– No false negatives *if for all $x \in \mathfrak{D}, \sigma \in \Sigma$ suach that $(\top, \sigma) \leftarrow \mathsf{ins}(x, \sigma)$, and for any sequence of insertions resulting in a later state σ', $b \leftarrow \mathsf{qry}(x, \sigma') \implies b = \top$.*

These rules are satisfied by, for example, Bloom and Cuckoo filters.

We use our new definition of *Ideal*, along with a modified version of the simulator from Fig. 4 that allows for permanent disabling. We manage to rederive results from [13], but with the NAI probabilities in the bounds being replaced with maximal NAI* probabilities. We note that this does not worsen the bounds for Bloom and Cuckoo filters.

Theorem 2. *Let q_{ins}, q_{qry} be non-negative integers, and let t_a, $t_d > 0$. Let $F\colon \mathfrak{D} \to \mathfrak{R}$. Let Π be an AMQ-PDS with public parameters pp and oracle access to F, such that Π satisfies F-decomposability (Def. 2), and reinsertion invariance (Def. 3). Let α, β be the number of calls to F required to insert or query an element respectively in Π using its $\mathsf{ins}, \mathsf{qry}$ algorithms.*

If Π satisfies permanent disabling *and* no false negatives *(Def. 15), $R_K\colon \mathfrak{D} \to \mathfrak{R}$ is an $(\alpha q_{ins} + \beta q_{qry}, t_a + t_d, \varepsilon)$-secure pseudorandom function with key $K \leftarrow_{\$} \mathcal{K}$, then Π is $(q_{ins}, q_{qry}, q_{del} = 0, t_a, t_s, t_d, \varepsilon')$-adversarially correct with respect to the simulator in Fig. 8, where $t_s \approx t_a$ and $\varepsilon' = \varepsilon + 2q_{qry} \cdot P^*_{\Pi,pp}(FP \mid q_{ins})$.*

Note that our result relies solely on *permanent disabling* and *no false negatives* (Def. 15). Focusing on NAI* instead of NAI probabilities allows the result not to depend on the other consistency rules, making it more general than the result in [13].

Proof. As in the proof of Theorem 1, we use an intermediate game G^* (Fig. 5 with the simulator from Fig. 8) to obtain

$$\begin{aligned}\mathrm{Adv}^{\mathrm{RoI}}_{\Pi,\mathcal{A},\mathcal{S}}(\mathcal{D}) &:= |\Pr[\,\mathit{Real}(\mathcal{A},\mathcal{D}){=}1\,] - \Pr[\,\mathit{Ideal}(\mathcal{A},\mathcal{D},\mathcal{S}){=}1\,]\,| \\ &\leq |\Pr[\mathit{Real}(\mathcal{A},\mathcal{D}){=}1] - \Pr[G^*(\mathcal{A},\mathcal{D}){=}1]| \\ &\quad + |\Pr[G^*(\mathcal{A},\mathcal{D}){=}1] - \Pr[\mathit{Ideal}(\mathcal{A},\mathcal{D},\mathcal{S}){=}1]|.\end{aligned}$$

Using the proof to Lemma 6 with $q_{\mathrm{del}} = 0$, we get a bound on the closeness of *Real*, G^* in terms of the PRF advantage ε, obtaining

$$\begin{aligned}\mathrm{Adv}^{\mathrm{RoI}}_{\Pi,\mathcal{A},\mathcal{S}}(\mathcal{D}) &:= |\Pr[\,\mathit{Real}(\mathcal{A},\mathcal{D}){=}1\,] - \Pr[\,\mathit{Ideal}(\mathcal{A},\mathcal{D},\mathcal{S}){=}1\,]\,| \\ &\leq \varepsilon + |\Pr[G^*(\mathcal{A},\mathcal{D}){=}1] - \Pr[\mathit{Ideal}(\mathcal{A},\mathcal{D},\mathcal{S}){=}1]|. \qquad (22)\end{aligned}$$

We next bound the closeness of G^*, *Ideal*, proving the statement.

Lemma 8. *The difference in probability of an arbitrary t_d-distinguisher $\mathcal{D}$ outputting 1 in experiments of game G^*-or-Ideal (Fig. 5 with the simulator from Fig. 8) with a $(q_{ins}, q_{qry}, q_{del} = 0, t_a)$-AMQ-PDS adversary $\mathcal{A}$ is bounded as:*

$$\begin{aligned}Adv^{G^*\text{-}or\text{-}Ideal}_{\Pi,\mathcal{A},\mathcal{S}}(\mathcal{D}) &:= |\Pr[\,G^*(\mathcal{A},\mathcal{D}) = 1\,] - \Pr[\,\mathit{Ideal}(\mathcal{A},\mathcal{D},\mathcal{S}) = 1\,]| \\ &\leq 2q_{qry} \cdot P^*_{\Pi,pp}(FP \mid q_{ins}).\end{aligned}$$

Proof. Let $\mathbf{E}$ be the divergence event between G^* and *Ideal*, which occurs due to a mismatch in responses to the adversary's queries across the two games (see Fig. 8). Then,

$$\Pr[\,\mathbf{E}\,] = \Pr[\,\mathbf{E_{Qry}}\,] \qquad (23)$$

with

$$\mathbf{E_{Qry}} := \left[\left[\begin{matrix}\text{The first query response mismatch is} \\ a_i^{Ideal} \neq a_i^{G^*} \text{ for some } i \in [q_{\mathrm{qry}}]\end{matrix}\right]\right], \qquad (24)$$

where a_i^G defines the response to the adversary's i-th query.

In contrast to the proof of Theorem 1 and its Eq. (5), Eq. (23) does not involve divergence events related to a mismatch in responses to **Del**, **Ins** queries across the two games. This is because the number of deletions is set to 0, allowing us to focus on the simulator from Fig. 8. Due to F-decomposability (Def. 2), both G^* and *Ideal* from Fig. 8 on insertions proceed to change the state in exactly the same way, outputting exactly the same answer to the adversary, and permanently disabling the structure at exactly the same time. So, responses to **Ins** queries cannot help the adversary to distinguish between the two games.

```
Simulator $\mathcal{S}(\mathcal{A}, pp)$
1 $F \leftarrow\!\!\$\ \mathsf{Funcs}[\mathfrak{D}, \mathfrak{R}]$
2 $\sigma \leftarrow \mathsf{setup}(pp)$
3 $\sigma^* \leftarrow \mathsf{setup}(pp)$
4 $\text{inserted} \leftarrow \{\}$
5 $f \leftarrow \{\}$
6 $\mathbf{return}\ \mathcal{A}^{\mathbf{InsSim},\ \mathbf{QrySim}}$

Oracle $\mathbf{InsSim}(x)$
1 $r \leftarrow\!\!\$\ \mathcal{R}$
2 $(c^{G^*}, \sigma^*) \leftarrow\!\!\$\ \mathsf{ins}^F(x, \sigma^*; r)$
3 $(c^{Ideal}, \sigma) \leftarrow\!\!\$\ \mathsf{ins}^{\mathrm{Id}_{\mathfrak{R}}}(F(x), \sigma; r)$
4 $\mathbf{return}\ c^{Ideal}$ $\boxed{\mathbf{return}\ c^{G^*}}$

Oracle $\mathbf{QrySim}(x)$
1 $a^{G^*} \leftarrow \mathsf{qry}^F(x, \sigma^*)$
2 $a^{Ideal} \leftarrow \top$
3 // If it isn't inserted
4 $\mathbf{if}\ \text{inserted}[x] = \bot$
5     $a^{Ideal} \leftarrow\!\!\$\ \mathsf{qry}^{\mathrm{Id}_{\mathfrak{R}}}(X \leftarrow\!\!\$\ \mathfrak{R}, \sigma)$
6 $\mathbf{return}\ a^{Ideal}$ $\boxed{\mathbf{return}\ a^{G^*}}$
```

Fig. 8: Simulator and $\boxed{G^*}$ for AMQ-PDS.

We now proceed to bound the probability of $\mathbf{E}_{\mathbf{Qry}}$. In the following, we take the probability over the randomness used by $\mathcal{A}$ (which we refer to as $\mathcal{A}$'s coins), and the randomness used by game $G \in \{G^*, \mathit{Ideal}\}$ to answer $\mathcal{A}$'s queries (which we refer to as G's coins). We will use x_i and z_i to denote the input to $\mathcal{A}$'s i-th query and insertion, respectively.

We first rewrite Eq. (2) using the union bound as

$$\Pr[\mathbf{E}_{\mathbf{Qry}}] \leq \sum_{i=1}^{q_{\mathrm{qry}}} \Pr\left[\begin{bmatrix}[\mathbf{Qry}(x_i)\text{ yields the first mismatch}]\wedge \\ [[(a_i^{Ideal} = \top) \wedge (a_i^{G^*} = \bot)] \\ \vee[(a_i^{Ideal} = \bot) \wedge (a_i^{G^*} = \top)]]\end{bmatrix}\right]. \tag{25}$$

In G^*, the responses to $\mathcal{A}$'s **Qry** queries are always computed using the same function F, while in *Ideal*, a fresh random string $X \leftarrow\!\!\$\ \mathfrak{R}$ is sampled each time a non-inserted element is queried.

Let σ_i denote the state of Π in game *Ideal* just before the i-th **Qry** query, and σ_i^* denote the corresponding state in game G^*. Due to the *no false negatives* consistency rule, σ_i^* has no false negatives. Moreover, **Qry** queries in *Ideal* do not give false negative responses. This means that **Qry** queries on elements that were inserted will always return a positive response in both games. Therefore, in order for x_i to yield a mismatch in the **Qry** query responses, we must have that x_i is not currently inserted in *Ideal* (i.e. inserted$[x_i] = \bot$ in line 4 of **QrySim**).

This gives

$$\Pr\left[\mathbf{E}_{\mathbf{Qry}}\right] \le \sum_{i=1}^{q_{\text{qry}}} \Bigg[\Pr\begin{bmatrix}[\mathbf{Qry}(x_i) \text{ yields the first mismatch }]\wedge \\ [\text{inserted}[x_i] = \bot] \wedge [a_i^{\mathit{Ideal}} = \top]\end{bmatrix}$$

$$+ \Pr\begin{bmatrix}[\mathbf{Qry}(x_i) \text{ yields the first mismatch }]\wedge \\ [\text{inserted}[x_i] = \bot] \wedge [a_i^{G^*} = \top]\end{bmatrix} \tag{26}$$

$$:= \sum_{i=1}^{q_{\text{qry}}} \Big[\Pr\left[\mathbf{E}_{\mathbf{Qry}}^{\mathit{Ideal}}\right] + \Pr\left[\mathbf{E}_{\mathbf{Qry}}^{G^*}\right] \Big], \tag{27}$$

where, for simplicity, we will use $\Pr\left[\mathbf{E}_{\mathbf{Qry}}^{\mathit{Ideal}}\right]$ to denote the first term of Eq. (26), and $\Pr\left[\mathbf{E}_{\mathbf{Qry}}^{G^*}\right]$ to denote the second term.

We start by bounding $\Pr\left[\mathbf{E}_{\mathbf{Qry}}^{\mathit{Ideal}}\right]$. In *Ideal*, a fresh random string $X \leftarrow_{\$} \mathfrak{R}$ is sampled each time a non-inserted element is queried, and so

$$\Pr\left[\mathbf{E}_{\mathbf{Qry}}^{\mathit{Ideal}}\right] \le \Pr_{\substack{\mathit{Ideal}\text{'s coins} \\ \mathcal{A}\text{'s coins}}} \begin{bmatrix}[\mathbf{Qry}(x_i) \text{ yields the first mismatch }]\wedge \\ [\top \leftarrow_{\$} \mathsf{qry}^{\mathrm{Id}_{\mathfrak{R}}}(X \leftarrow_{\$} \mathfrak{R}, \sigma_i)]\end{bmatrix}. \tag{28}$$

We now argue that every σ_i is an n-NAI* state, with n being upper bounded by q_{ins}, by showing that it satisfies the requirements in Corollary 1. Let $\{z_j\}_{j\in[j_i]}$ be inputs to $\mathcal{A}$'s insertions before the i-th query.

As $\mathcal{A}$ has no direct access to random function F and $\mathit{\Pi}$ satisfies *permanent disabling*, no information about $F(z_j)$ is available to $\mathcal{A}$ before z_j has been successfully inserted. Unsuccessful insertions do not reveal any information about the action of F on elements not yet inserted. This is because if an insertion does not succeed, *permanent disabling* ensures that also no other element can be successfully inserted. So, every first call to **InsSim** on z_j can use $X_{z_j} \leftarrow_{\$} \mathfrak{R}$ in place of $F(z_j)$, without changing the distribution of the resulting state. Therefore, σ_i^* satisfies *insertion unpredictability* (Def. 6).

The requirements of Corollary 1 related to deletions are trivially satisfied as $q_{\text{del}} = 0$. Moreover, reinsertions do not change the state due to *reinsertion invariance.*

Thus, σ_i is an NAI* state containing at most q_{ins} elements, and we can upper bound the false positive probability of σ_i by that of the NAI* state with the maximal false positive probability (Def. 8), giving

$$\Pr\left[\mathbf{E}_{\mathbf{Qry}}^{\mathit{Ideal}}\right] \le \Pr_{\substack{\mathit{Ideal}\text{'s coins} \\ \mathcal{A}\text{'s coins}}} \begin{bmatrix}[\mathbf{Qry}(x_i) \text{ yields the first mismatch }]\wedge \\ [\top \leftarrow_{\$} \mathsf{qry}^{\mathrm{Id}_{\mathfrak{R}}}(X \leftarrow_{\$} \mathfrak{R}, \sigma_i)]\end{bmatrix}$$

$$\le P^*_{\mathit{\Pi},pp}(FP \mid q_{\text{ins}}). \tag{29}$$

We next compute

$$\Pr\left[\mathbf{E}_{\mathbf{Qry}}^{G^*}\right] = \Pr_{\substack{G^*\text{'s coins} \\ \mathit{Ideal}\text{'s coins} \\ \mathcal{A}\text{'s coins}}} \begin{bmatrix}[\mathbf{Qry}(x_i) \text{ yields the first mismatch }]\wedge \\ [\text{inserted}[x_i] = \bot] \wedge [\top \leftarrow_{\$} \mathsf{qry}^{F}(x_i, \sigma_i^*)]\end{bmatrix}. \tag{30}$$

We have that a_i^{G*} is never returned to $\mathcal{A}$ in *Ideal*, $\mathcal{A}$ has no direct access to random function F and $\mathit{\Pi}$ satisfies *permanent disabling*. So, no information about $F(x_i)$ is available to $\mathcal{A}$ (as x_i has never been successfully inserted), and we can write Eq. (30) as

$$\Pr\left[\mathbf{E}_{\mathbf{Qry}}^{G^*}\right] \leq \Pr_{\substack{\mathit{Ideal}\text{'s coins} \\ \mathcal{A}\text{'s coins}}}\left[\begin{matrix}[\mathbf{Qry}(x_i) \text{ yields the first mismatch }]\wedge \\ [\top \xleftarrow{\$} \mathsf{qry}^{\mathrm{Id}_{\mathfrak{R}}}(X_i \xleftarrow{\$} \mathfrak{R}, \sigma_i^*)]\end{matrix}\right]. \tag{31}$$

Since $\sigma_i^* = \sigma_i$ (Fig. 8), the right-hand sides of Eq. (31) and Eq. (28) are equal, and Eq. (29) implies

$$\Pr\left[\mathbf{E}_{\mathbf{Qry}}^{G^*}\right] \leq P_{\mathit{\Pi},pp}^*(FP \mid q_{\mathrm{ins}}). \tag{32}$$

Substituting $\Pr\left[\mathbf{E}_{\mathbf{Qry}}^{\mathit{Ideal}}\right], \Pr\left[\mathbf{E}_{\mathbf{Qry}}^{G^*}\right]$ in Eq. (27) gives

$$\Pr\left[\mathbf{E}_{\mathbf{Qry}}\right] \leq \sum_{i=1}^{q_{\mathrm{qry}}} 2P_{\mathit{\Pi},pp}^*(FP \mid q_{\mathrm{ins}}) = 2q_{\mathrm{qry}} \cdot P_{\mathit{\Pi},pp}^*(FP \mid q_{\mathrm{ins}}). \tag{33}$$

□

To prove Theorem 2, we then apply Lemma 8 to Eq. (22) and obtain

$$\mathrm{Adv}_{\mathit{\Pi},\mathcal{A},\mathcal{S}}^{\mathrm{RoI}}(\mathcal{D}) \leq \varepsilon + 2q_{\mathrm{qry}} \cdot P_{\mathit{\Pi},pp}^*(FP \mid q_{\mathrm{ins}}).$$

□