# Bet-or-Pass: Adversarially Robust Bloom Filters[⋆]

Moni Naor and Noa Oved

Department of Computer Science and Applied Mathematics, Weizmann Institute of Science, Rehovot, Israel {moni.naor, noa.oved}@weizmann.ac.il

**Abstract.** A *Bloom filter* is a data structure that maintains a succinct and probabilistic representation of a set $S \subseteq U$ of elements from a universe $U$. It supports approximate membership queries. The price of the succinctness is allowing some error, namely false positives: for any $x \notin S$, it might answer 'Yes' but with a small (*non-negligible*) probability.

When dealing with such data structures in adversarial settings, we need to define the correctness guarantee and formalize the requirement that bad events happen infrequently and those false positives are appropriately distributed. Recently, several papers investigated this topic, suggesting different robustness definitions.

In this work we unify this line of research and propose several robustness notions for Bloom filters that allow the adaptivity of queries. The goal is that a robust Bloom filter should behave like a random biased coin even against an adaptive adversary. The robustness definitions are expressed by the type of test that the Bloom filter should withstand. We explore the relationships between these notions and highlight the notion of *Bet-or-Pass* as capturing the desired properties of such a data structure.

## 1 Introduction

A *Bloom filter* is a data structure that maintains a succinct representation of a set $S \subseteq U$ of elements from a universe $U$. It supports an approximate version of membership queries: for any $x \in S$, the Bloom filter must answer 'Yes' while for any $x \in U \setminus S$, it should answer 'No', but is allowed to have a small error probability[1] (at most $\varepsilon$), and answer 'Yes'. That is, it admits false positives but not false negatives.

The small memory required by the construction of Bloom filters (as opposed to storing $S$ precisely) and the fast query time make Bloom filters extremely attractive in various applications. This comes at the price of a certain rate of false positive - elements not in the set declared as being in the set. False positives can affect performance, e.g., they can incur unnecessary disk access, lead to spam emails that are not marked as spam, and allow misspelled words. Therefore, the

[⋆] Research supported in part by grants from the Israel Science Foundation (no.2686/20), by the Simons Foundation Collaboration on the Theory of Algorithmic Fairness. Moni Naor is the incumbent of the Judith Kleeman Professorial Chair.

[1] The precise meaning of this probability is the subject of this paper.

false positive rate is the main correctness metric that interests us. It is important to note that the false positive rate cannot be negligible if we wish to save space[2].

When we are dealing with a data structure such as Bloom filters, with a *non-negligible* false-positive rate the question is how should we define the correctness guarantee: what does it mean to see bad events infrequently, i.e., how can we claim the data structure behaves "nicely". (In contrast, in most of cryptography a "bad" event happens with only negligible probability and the definition of security is that we aren't likely to see it at all). One way to define the correctness is by first fixing a sequence of inputs (equivalently, the queries) and then show an upper bound on the false positive rate. However, this is not sufficient in many scenarios, especially when the queries are chosen adaptively, based on previous queries' responses.

This work proposes several robustness notions for Bloom filters that allow adaptivity and capture adversaries with different goals, using different evaluation metrics. The robustness definitions are formalized as tests to the Bloom filter. We investigate the relationships between these notions and propose one notion as the most desirable one to define robust Bloom filters.

There are many variants of Bloom filters, for instance, where the Bloom filter is initially empty and the set $S$ is defined via insert queries, or where some extra information is attached to each element and we wish to retrieve this information in case the element is in the set. In this work we concentrate on the case where the set $S$ is fixed and the queries are adaptively chosen. Our definitions are relevant to the other variants as well, and as far as we can see, so are the relationships we found. In addition, the question of defining the resiliency of a data structure with non-negligible failure faced with an adaptive adversary is relevant to other data structures, and our results may apply to them as well.

*Robust Bloom filters.* The correctness of Bloom filters was mainly analyzed under the assumption that we first fix a query $x$ and then compute the error probability over the internal randomness. We refer to this as the *static analysis*. One might ask what happens when an adversary chooses the next query based on the response of previous ones? Does the error probability remain the same? Those questions motivated the analysis of Bloom filters in adversarial settings, where an adversary chooses her queries adaptively.

We refer to a Bloom filter as robust if it satisfies some correctness guarantee under adaptive adversarial settings. Our *wishful thinking* is that a "robust" Bloom filter should behave like a truly unpredictable biased coin; that is, each query is false positive with probability at most $\varepsilon$ regardless of the result of previous queries. Indeed, this is the case in the static settings. However, it is not true and more complex to formalize when considering a sequence of (mostly adaptively) chosen queries. One reason this is not true is that seeing the response on previous inputs might leak some information about the internal state of the data structure

[2] The lower bound on memory requirements of a Bloom filter is $n \log 1/\varepsilon$ where $n$ is the size of the set $S$ and $\varepsilon$ is the error probability.

or the random bits used. This, in turn, can be used by an adversary that, for example, wants to increase the false-positive rate.

An example to demonstrate an adaptive attack is when using Bloom filters in Web cache sharing (see [12]). When a proxy gets a request for a web page, it first checks if the page is available in its cache, and only then does it search for the web page on another proxy cache. As a final resort, it requests the web page from the Web. Therefore, proxies must know the cache's content of other proxies. For such a scheme to be effective, proxies do not transfer the exact contents of their caches but instead periodically broadcast Bloom filters that represent it. If a proxy wants to know if another proxy has a page in its cache, it checks the corresponding Bloom filter. In case of false positives, a proxy may request a page from another proxy, only to find that this proxy does not have that page. In this case, a delay is caused. In the static analysis, one would set the error to be small such that cache misses rarely happen. However, an adversary requesting for web pages can time the result of the proxy, and learn the responses of the Bloom filters. In turn, this might enable her to find false positives and cause unsuccessful cache access, which leads to an overload. Note that the adversary cannot repeat a false positive since the proxy will save it in its cache once a web page is requested. A similar example was presented in [23].

### 1.1 Our Contributions

We explore old and new notions of *robustness* for Bloom filters and study the relationships between them. The precise definitions are given in Section 3. Our definitions aim to capture the idea that a robust Bloom filter should behave like a random biased coin even for an adaptive adversary. We highlight the notion of *Bet-or-Pass* as capturing the desired properties of such a data structure. First, as we shall see, it gives us the strongest guarantee we can (currently) imagine. Second, it is not too strong: there is a Bloom filter satisfying this notion (one based on a construction in [23]). Finally, it is relatively convenient to check whether a suggested construction of a Bloom filter satisfies the definition.

Following the work of Naor and Yogev [23] we define robustness tests in the form of a game with an adversary. The adversary chooses the set $S$ and adaptively queries the Bloom filter. The goal of the adversary differs between tests. Naor and Yogev defined that following the adaptive queries, the adversary **must** output a never-queried before element $x^*$, which she thinks is a false positive. They said that an adversary wins if $x^*$ is a false positive. They wanted the probability of an adversary to win (equivalently- make the Bloom filter fail the test) to be at most $\varepsilon$. We refer to the security notion of Naor and Yogev as the *Always-Bet (AB) test*.

We define a new test, extending the AB test. First, we allow the adversary to *pass*, meaning she does not have to provide any output. This gives the adversary more flexibility and defines a more robust test. In addition, we define an adversary's profit: if she outputs (bets on) an element $x^*$ which is indeed a false positive, she is rewarded; otherwise, she "pays". If she chooses to pass, her profit is zero. Our profit definition gives rise to a new metric to evaluate Bloom filters: we set the payments so that a random guess with probability $\varepsilon$ has an expected profit of 0. We say that an adversary makes the Bloom filter fail in the *Bet-or-Pass (BP) test* if her expected profit is noticeably larger than 0. In this case, the Bloom filter is not BP test resilient, (not robust under the BP test).

The AB and the BP tests consider a one-time challenge, $x^*$. We also consider tests with a "continuous" flavor; those tests examine the false positive rate in the entire sequence of adaptive queries and look for "anomalies". We propose a new family of tests following our original desire to require a robust Bloom filter to behave like a truly unpredictable biased coin. Informally, it tests whether a sequence generated by the output of a Bloom filter on adaptively selected queries "looks like" a biased random coin to any efficient observer (that examines some property of the sequence). Since there can be elements that are always true negatives and we are only interested in cases where an adversary *increases* the false positive rate, we consider *monotone observers* only - observers that test a monotone property of the sequence. In other words, observers that are sensitive to the addition of false positives and not the reduction of ones. If a Bloom filter does not fail in all monotone tests, we say it is *monotone test* resilient. We then analyze a special case of the monotone test: we look at the expected number of false positives. We say that a Bloom filter is *expected count test* resilient if for all adversaries the expected number of false positive in $t$ queries is at most the expected number of ones in a sequence of $t$ independent biased coin tosses.

Finally, we emphasize why adaptive queries are interesting by introducing a test we call *the semi-adaptive prediction test.* In this test, the adversary commits to a set of queries $Q$ (non-adaptive) before getting access to query the Bloom filter. The adversary aims to find a false positive element from $Q$ that was not queried yet (the adaptive part). A Bloom filter is a semi-adaptive prediction resilient if no adversary can find a false positive element from $Q$ with a probability of at least $\varepsilon$.

*Relationships.* We explore the relationships between the different definitions (see Fig. 1). We prove that a Bloom filter that is BP test resilient is also AB test resilient. On the other hand, we show that a Bloom filter that is AB test resilient is not necessarily BP test resilient. This suggests that the BP test is a more robust notion than the AB test. We support this idea by showing that *BP test resilience implies monotone test resilience*[3] while a Bloom filter that is AB test resilient is not necessarily monotone test resilient. However, we show that AB test resilience, in turn, implies expected count test and semi-adaptive prediction resilience. We also demonstrate that the expected count test and semi-adaptive prediction are weak notions: we construct Bloom filters that satisfy those notions and fail the AB test. Finally, we show that monotone test resilience implies expected count test resilience, supporting that the expected count is indeed a special case of monotone test. We conclude that being resilient to the BP test guarantees the

[3] This is reminiscent of the fact that in pseudorandomness the next-bit-test implies all efficient tests.

desired robust properties and suggests it is the correct way to define a robust Bloom filter.

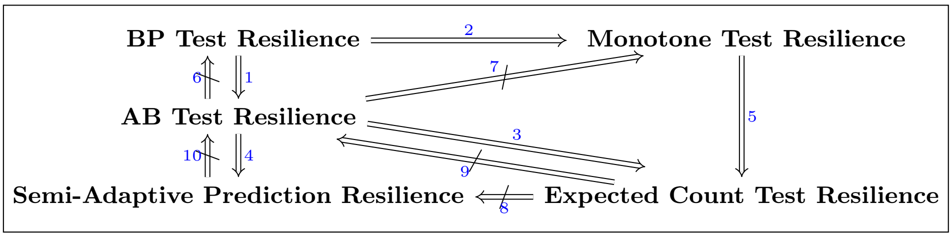

**Fig. 1.** The relationships between the different definitions.

*BP as a natural notion of robustness.* Does the BP test capture the desired behavior of a Bloom filter as a random (biased) coin? At first glance, the skeptical reader might think that the option of passing is strange[4]. However, given that BP test resilience implies monotone test resilience, we can get some intuition why this is indeed the case. The idea is that when a Bloom filter fails in the monotone test, we can use the monotone distinguisher to know when to bet. This means that when a Bloom filter's behavior is distinguishable from a random coin, an adversary can exploit that to guess a false positive element with a high probability (higher than a random guess).

Our work explores what it means to be a robust Bloom filter. When trying to suggest a definition, we aim to achieve three requirements: First, the definition is sufficient, i.e., it captures the idea that a robust Bloom filter should behave like a random biased coin. Second, it's not too strong, i.e., there exists a construction of a Bloom filter satisfying this property. Lastly, it is easy to use, i.e., it is formalized as a **simple** test for a Bloom filter. The BP test resilience definition satisfies **all** these requirements, suggesting it is a natural notion of robustness.

Finally, we give an example to directly motivate the BP definition. Consider a system containing $k$ different components, each using a Bloom filter to store some set (e.g., $k$ web proxies with a Bloom filter holding their cache content). Further, assume that the system as a whole can withstand a certain false positive rate, denoted by epsilon. Suppose that there exists an adversary that, with noticeable probability, can find a false positive element with high probability (greater than epsilon) and knows to indicate when it happens. We can use this adversary in all the $k$ components simultaneously in the following way: we locate $k'$ components with a corresponding false positive element. We then query all those $k'$ elements at about the same time. This results in a short period with a high false positive

[4] An analogy is situation in the casino game of blackjack, where at a certain point in the game the participants may have a small advantage over the house, as more cards are exposed (followed via "card counting"), and may choose to start betting then or to increase their bets.

rate in the system, which can cause a denial of service. Note that the other notions fail to capture this attack.

*The surprise exam and the AB and BP notions.* We can demonstrate the difference between the AB and BP notions by thinking of these variants wrt the famous surprise exam (or unexpected hanging) paradox [11]. Suppose a teacher announces that a "surprise exam" will occur sometime in the next six days and choose the date at random. On the evening of each day, a student can bet whether the exam will happen the next day. If the student is correct, then she wins 5 dollars; if she is wrong, she loses a Dollar. In both settings the student can bet only once. In the AB setting, she must bet in at least one day, and in the BP setting, she can decide not to bet at all. The expected value in the AB setting is 0 (against a random day). In contrast, in the BP setting, the student can wait until the last day and bet only if the exam did not take place before. In this case, she **knows** the exam is on the last day and has a strictly positive expectation (5/6). If the teacher does not chooses a day at random, but with some other distribution, then we need a more sophisticated strategy, but it is doable.

*Computational Assumptions and One-way Functions.* Naor and Yogev [23] proved existential equivalence between Bloom filters that are AB test resilient (against a computationally bounded adversary) and one-way functions[5]. We refer to it as the equivalence result. We ask whether this equivalence still holds given a Bloom filter that is BP test resilient. The simpler direction shows that a Bloom filter that is BP test resilient implies the existence of one way function (we get it immediately by the implication of the BP test on the AB test). Showing the other direction is a little bit more challenging. We show a modification of the construction of Bloom filter from [23] that is based on the existence of one-way functions and prove it is BP test resilient. This, in turn, show the desired equivalence.

In the full version of this paper, we also ask whether weaker notions of robustness imply one-way functions. We show that if one-way functions do not exist, then any non-trivial[6] Bloom filter fails the expected count test the semi-adaptive prediction test.

### 1.2 Related work

The first work to consider adaptive adversaries that choose queries based on the response of the Bloom filter is by Naor and Yogev [23]. They defined an adversarial model for Bloom filters through a game with an adversary. The adversary has only oracle access to the Bloom filter and cannot see its internal randomness. She can adaptively query the filter, and her goal is to find a never-queried-before false-positive element. We continue this line of research by introducing new adversarial models, suggesting new ways to evaluate the Bloom filter performance.

[5] One-way functions are functions that, informally speaking, are easy to compute but hard to invert.

[6] Non-trivial Bloom filters are Bloom filters that require less space than the amount of space required to explicitly store the set.

Naor and Yogev also presented a tight connection between Bloom filters in their model and one-way functions, which we extend to our settings.

Following [23], Clayton, Patton, and Shrimpton [8] analyzed Bloom filters, as well as other data structures such as counting Bloom filters and count-min sketches, in adversarial settings. They analyzed the probability of getting some predefined number of false-positive elements in a sequence of adaptive queries (as opposed to the probability of finding one never-queried before false-positive element). This type of analysis is similar to our expected count test.

Another move towards adaptivity was made by Bender et al. [2]. Similarly to [23], they indicated that the bound of false-positive probability only applies to a single fixed query, and a sequence of queries can have a much larger false positive rate (simply by repeating a false positive query). Their main concern was when an adversary *repeats* a false positive query (unlike Naor and Yogev, which did not allow repeating queries). To deal with this type of attack, they defined an *adaptive filter*: a filter that adapts to false positives, which means that even for an element that was queried and returned as a false positive, repeating it results in a false positive rate of at most $\varepsilon$. Their analysis assumes that the adversary could not find a never-queried-before element that is a false positive with probability greater than $\varepsilon$ when using the result of previous queries. Their assumption can be achieved using the constructions in [23]. Therefore, their work is orthogonal to Naor and Yogev (and ours), since their concern is dealing with repeated queries and does not handle the issue of using adaptivity to find never-queried false positives. Repeated queries were also discussed by Mitzenmacher et al. [21] (adaptive cuckoo filter), and by Lee et al. [18] (telescoping adaptive filter (TAF)).

The problem of defining correctness in adaptive settings was also investigated in the streaming algorithms literature, where there is a growing interest in *adversarial* streaming algorithms. These algorithms preserve their efficiency and correctness even if the stream is chosen adaptively by an adversary that observes the algorithm's output (and therefore can depend on the internal randomness of the algorithm). Hardt and Woodruff [15] showed that linear sketches are inherently non-robust to adaptively chosen inputs and cannot be used to compute the Euclidean norm of its input (while they are primarily used for this reason in the static setting). Kaplan et al. [17] introduced a streaming problem that shows a gap between adversarial and oblivious streaming in the space complexity requirement. On the positive side, Ben-Eliezer et al. [1] presented generic compilers that transform a non-robust streaming algorithm into a robust one in various scenarios. Hassidim et al. [16] and Woodruff and Zhou [28] continued their work suggesting better overhead.

### 1.3 Open Problems

Our work leaves open several interesting directions. The direct one is whether Monotone Test Resilience implies BP Test Resilience (recall that we show the other direction), or whether the two notions are separable. Are the techniques of Bender et al. [2] compatible with our results? Namely, can we have a robust BP secure Bloom filter against repetition? Another interesting avenue concerns the idea of allowing to output “don't know” (pass). To better capture the bit-security of decision games, Micciancio and Walter permitted an adversary to output a bot and redefined the advantage of the adversary conditioning on not outputting a bot (Def. 9 in [20]). Outputting a bot is very similar to pass in the BP test, and allowing pass and redefining the advantage gives a better notion. One wonders if there is a connection between these two definitions and if our definition can be applied in a more general setting (e.g., defining better bit-security for general decision games). It seems that knowing you don't know is significant in both cases. The definitions are related but not identical. For instance, in our case, the probability of the "secret bit" being 0 or 1 does not equal 1/2 (the probability of it being 1, a false positive element, is $\epsilon$). In addition, our game is asymmetrical: there is a difference between the bits 0 and 1 (we are only interested in the adversary outputting 1). We believe that it can be applied in a more general setting, but more work needs to be done to determine that.

## 2 Model and Problem Definition

For a universe $U = [u]$ we are given a subset $S \subset U$ of $n$ elements. The set can either be fixed throughout the lifetime of the Bloom filter or can be formed via insert queries. As mentioned, we consider in this work the case where the set $S$ is fixed; however, our results can be extended to other settings.

Following the work of [23], we model a Bloom filter $\mathbf{B} = (\mathbf{B}_1, \mathbf{B}_2)$ as a data structure consisting of two parts: a setup algorithm $\mathbf{B}_1$ and a query algorithm $\mathbf{B}_2$. The setup algorithm is randomized, gets a set $S$ as input, and outputs a compressed representation of $S$, denoted by $M$. The query algorithm $\mathbf{B}_2$, can be randomized, is given a compressed representation of a set $S$, and answers membership queries. It gets an element $x \in U$ and outputs 0 or 1, indicating whether $x$ belongs to $S$ or not (and may be wrong for $x \notin S$). For simplicity of notation, we consider a probabilistic query algorithm that cannot change the set representation. However, our results also apply to Bloom filters that can change the set representation.

If $x \notin S$ and $\mathbf{B}_2(\mathbf{B}_1(S), x) = 1$ we say that $x$ is a *false positive*. The main evaluation metric of a Bloom filter is the false positive rate.

**Definition 1 (Bloom filter).** *Let $\boldsymbol{B} = (\boldsymbol{B}_1, \boldsymbol{B}_2)$ be a pair of probabilistic polynomial time algorithms such that $\boldsymbol{B}_1$ gets as input a set $S$ and outputs a representation $M$, and $\boldsymbol{B}_2$ gets as input a representation $M$ and a query element $x \in U$ and outputs a response to the query. We say that $\boldsymbol{B}$ is an $(n, \varepsilon)$-Bloom filter if for all sets $S$ of size $n$ in a suitable universe $U$ it holds that:*

1. *Completeness: For any $x \in S$ we have that* $\Pr[\boldsymbol{B}_2(\boldsymbol{B}_1(S, x)) = 1] = 1$
2. *Soundness: For any $x \notin S$ we have that* $\Pr[\boldsymbol{B}_2(\boldsymbol{B}_1(S, x)) = 1] \leq \varepsilon$*,*

*where the probabilities are over the setup algorithm $\boldsymbol{B}_1$ and query algorithm $\boldsymbol{B}_2$.*

From here on, we always assume that $\mathbf{B}$ has this format and sometimes write $\mathbf{B}(S, x)$ instead of $\mathbf{B}_2(\mathbf{B}_1(S), x)$.

*The Adaptive Game.* Def. 1 considers a single fixed input element $x$, and the probability is taken over the randomness of the Bloom filter (and not over the choice of $x$, for example). This is a weak guarantee that we want to strengthen. We consider a sequence of $t$ inputs $x_1, \ldots, x_t$ that is not fixed but chosen adaptively by an adversary.

Defining adaptivity requires specifying what information is made available to the adversary to adapt. Here, we allow the adversary to see the responses of previous queries before choosing the next one. We formalize this by defining a game $\mathsf{AdaptiveGame}_{\mathcal{A},t}(\lambda)$ where $\lambda$ is a security parameter (see below). In this game, we consider a polynomial-time adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ that consists of two parts: $\mathcal{A}_1$ chooses the set $S$, and $\mathcal{A}_2$ gets as input the set $S$ and oracle access to the query algorithm (initialized with $M$) and perform adaptive queries. $\mathcal{A}_2$ aims to achieve a different goal in each robustness definition in order to make the Bloom filter fail the game (equivalently, the test). We measure the ability of $\mathcal{A}$ to make the Bloom filter fail with respect to her and the Bloom filter randomness.

To handle a computationally bounded adversary, we add a *security parameter* $\lambda$, which is given to the setup phase of the Bloom filter and the adversary as an input (acts as a key length). We now view the running time of the adversary, as well as her probability to make the Bloom filter fail, as functions of $\lambda$. Moreover, it enables the running time of the Bloom filter to be polynomial in $\lambda$ and hence the false positive probability $\varepsilon$ can be a function of $\lambda$.

We use the notation $\mathsf{negl}$ for any function $\mathsf{negl}\colon \mathbb{N} \to \mathbb{R}^+$ satisfying that for every positive polynomial $p(\cdot)$ there is an $N$ such that for all integers $n > N$ it holds that $\mathsf{negl}(n) < \frac{1}{p(n)}$. Such functions are called *negligible.*

**Definition 2.** ***The adaptive game*** $\mathsf{AdaptiveGame}_{\mathcal{A},t}(\lambda)$***:***

1. *The adversary $\mathcal{A}_1$ is given input $1^{\lambda+n\log u}$ and outputs a set $S \subset U$ of size $n$.*
2. *$\boldsymbol{B}_1$ is given input $\left(1^{\lambda+n\log u}, S\right)$ and builds a representation $M$.*
3. *The adversary $\mathcal{A}_2$ is given input $\left(1^{\lambda+n\log u}, S\right)$ and oracle access to $\boldsymbol{B}_2(M,\cdot)$ and performs at most $t$ adaptive queries $x_1, \ldots, x_t$ to $\boldsymbol{B}_2(M,\cdot)$.*

We assume wlog that $x_i \notin S$ for all $i \in [t]$, since Bloom filters admit false positives, but not false negatives, and also, $\mathcal{A}_2$ is given as input the set $S$. Therefore a queried element can be either false positive or true negative.

**Definition 3 (A test for Bloom filters).** *Let $\boldsymbol{B}$ be an $(n,\varepsilon)$-Bloom filter. For a security parameter $\lambda$, an $(n,t,\varepsilon)$-$\mathsf{Test}(\lambda)_{\mathcal{A},t}$ start with an adaptive game $\mathsf{AdaptiveGame}_{\mathcal{A},t}(\lambda)$ with an adversary $\mathcal{A}$. The test is defined by a function that given the transcript of the game (including the set $S$) decides 'succeed' or 'fail'.*

We say that a Bloom filter is *resilient for some test* (or family of tests) if the probability that any adversary makes it fail the test is upper bounded by some value. This term is formalized for each test in Section 3.

*The role of $t$.* The parameter $t$ indicates the number of queries the adversary performs. When $t$ is not known in advance and unbounded, the adversaries must be computationally bounded given the equivalence result of [23] (see Section 1.1). However, when $t$ is known in advance, the adversary does not have to be computationally bounded: Naor and Yogev presented a construction of a Bloom filter that is AB test resilient against computationally *unbounded* adversary using $O\left(n \log \frac{1}{\varepsilon} + t\right)$ bits of memory.

Inspired by Def. 2.5 in [23] we say that if **B** is resilient for any polynomial number of queries, it is *strongly resilient.*

**Definition 4 (Strongly resilient).** *For a security parameter $\lambda$, we say that $\boldsymbol{B}$ is an $(n, \varepsilon)$-strongly $\mathsf{Test}(\lambda)_{\mathcal{A},t}$ resilient, if for any polynomial $p(\cdot)$ and $t \le p(\lambda, n)$ it holds that $\boldsymbol{B}$ is an $(n, t, \varepsilon)$-$\mathsf{Test}(\lambda)_{\mathcal{A},t}$ resilient.*

An essential property of a Bloom filter is its memory size. Bloom filters are used because their memory size is smaller than an explicit representation of the set. We say that a Bloom filter uses $m$ bits of memory if the largest representation for all sets $S$ of size $n$ is at most $m$. Carter et al. [7] showed that in order to construct a Bloom filter for sets of size $n$ and error rate $\varepsilon$ one must use (roughly) $m \ge n \log \frac{1}{\varepsilon}$ bits of memory (as opposed to $n \log u$ bits needed to answer **exact** membership queries). We can write this as $\varepsilon \ge 2^{-\frac{m}{n}}$ which leads us to the following definition:

**Definition 5 (Minimal error (Def. 2.7 in [23])).** *Let $\boldsymbol{B}$ be an $(n, \varepsilon)$-Bloom filter that uses $m$ bits of memory. We say that $\varepsilon_0 = 2^{-\frac{m}{n}}$ is the minimal error of $\boldsymbol{B}$.*

A simple construction of a robust Bloom filter can be achieved by storing $S$ precisely, and then there are no false positives for an adversary to find. The disadvantage of this solution is that it requires a large memory, while Bloom filters aim to reduce the memory size. Similarly, a Bloom filter with a substantially low false-positives rate is robust. We are interested in a robust non-trivial Bloom filter. Roughly speaking, a non-trivial Bloom filter is a Bloom filter with $\varepsilon$ substantially far from 0 and 1 and a large universe (compared to the memory size, so it will not be possible to store the set explicitly). For convenience, we use the definition of [23].

**Definition 6 (Non-trivial Bloom filter (Def. 2.8 in [23])).** *Let $\boldsymbol{B}$ be an $(n, \varepsilon)$-Bloom filter that uses $m$ bits of memory and let $\varepsilon_0$ be the minimal error of $\boldsymbol{B}$. We say that $\boldsymbol{B}$ is non-trivial if for all constants $a > 0$ it holds that $u > \frac{a \cdot m}{\varepsilon_0^2}$ and there exists polynomials $p_1(\cdot), p_2(\cdot)$ such that $\frac{1}{p_1(n)} < \varepsilon_0 \le \varepsilon < 1 - \frac{1}{p_2(n)}$.*[7]

[7] If $\varepsilon$ is negligible in $n$, then any polynomial-time adversary has only a negligible chance of finding any false positive. In that case, we can transform any adaptive adversary into a non-adaptive adversary since it knows the answers already. The same argument appears in [6] as Lemma 4. A similar claim applies to the requirement that $\varepsilon$ will be substantially far from 1.

Let $\lambda$ be the security parameter. It holds that $n = poly(\lambda)$.[8] Therefore we get that $\varepsilon_0$ cannot be too small and $\varepsilon$ cannot be too large. I.e.

$$\frac{1}{q_1(\lambda)} \leq \varepsilon_0 \leq \varepsilon \leq 1 - \frac{1}{q_2(\lambda)}, \tag{1}$$

for some polynomials $q_1(\cdot)$ and $q_2(\cdot)$.

## 2.1 Prediction and Pseudorandomness

Pseudorandomness captures the idea that an object can "look" completely random even though it is far from random, since it was generated from a much shorter seed. One approach to defining it formally is via *indistinguishability*: Let $\mathsf{Dist}$ be a distribution on $t$-bit strings. We say that $\mathsf{Dist}$ is pseudorandom if it is infeasible for any polynomial-time algorithm to *distinguish* (in a non negligibly better way than guessing) whether it is given a string sampled according to $\mathsf{Dist}$ or whether it is given a uniform $t$-bit string. This means that pseudorandomness is a computational relaxation of true randomness.

The definition above introduces many tests: each polynomial-time algorithm (distinguisher) serves as a single test. An example of such a test is an algorithm $D$ that returns '1' if the input string's first bit is 0. Therefore, the first bit of a string sampled from $\mathsf{Dist}$ should be equal to 0 with probability very close to 1/2, since the first bit of a string sampled from a uniform distribution equals 0 with probability exactly 1/2. Otherwise, $D$ can distinguish between the two distributions.

An alternative approach to defining pseudorandomness is having a **single** type of test, *the next bit test*. In this test, a probabilistic polynomial-time algorithm is given a prefix of bits. It aims to *predict* the next bit of the source with a probability of success significantly greater than 1/2 (this is the Blum-Micali definition [5]).

As Yao [29] showed, these two definitions are equivalent and the infeasability of predicting some bit of a given source can serve as a test for randomness. We show something of a similar nature. Consider a Bloom filter, an attacker and the sequence of bits representing whether a false positive occurred or not on a sequence of queries. Since the probability of a false positive is at most $\varepsilon$, we deal with a non-uniform output distribution. We want this distribution to be indistinguishable from a random independent biased sequence. This means that the false positive events are random independent events even in the case of adaptive queries. We formalize it in the monotone test.

Similar to the result in pseudorandomness, we want to define a prediction test that implies the monotone test. Our starting point is the natural extension of the next bit test for biased bits. This test requires that no observer succeeds in predicting the bits of the source with a probability greater than the bias. However, Schrift and Shamir [27] showed that the natural extension is no longer

[8] Since we want the adversary to run in polynomial time in the security parameter.

a universal test for independence. Therefore, we choose a different approach that draws inspiration from the world of gambling.

There are tight connections between gambling and knowledge of a sequence. Cover and Thomas ([9] Chapter 6) used gambling to express the ability to predict. They investigated the relationship between information theory and gambling. They looked at a horse race and presented two equivalent ways to describe the odds. We show one of them, which is referred to as $b$−to−1: the gambler will pay 1 dollar after the race if his horse loses, and pick up $b$ dollars after the race if his horse wins. They looked at the gambler's wealth, which he wishes to maximize. We use their gambling methodology to define a robust Bloom filter. We let an adversary bet on a false positive: she outputs an element that she believes is a false positive. She gets rewarded if her output element is a false positive, while she is penalized if she outputs a true negative. We allow the adversary to pass, meaning she does not have to bet. In this case, she does not gain or lose any value. She wishes to maximize her wealth as well. We formalize it in the Bet-or-Pass Test and show it implies the monotone test.

# 3 Defining Robust Bloom filters

## 3.1 Background

We have a data structure, Bloom filter, with a non-negligible rate of false positives, denoted by $\varepsilon$. We want to claim it performs well. We can think of the Bloom filter response to a sequence of queries as a sequence of independently biased coin tosses (with bias $\varepsilon$ to 1). This is mostly the case when an adversary performs non-adaptive queries; she chooses her queries without seeing the response of the Bloom filter on previous queries. In that case, she gets a false positive (equivalently, one as a response) with probability at most $\varepsilon$ in each query. Our *wishful thinking* is that a Bloom filter behaves like a truly unpredictable biased coin even when an adversary sees the response of the Bloom filter on previous queries. Meaning it performs well (robust) even in adaptive settings. However, this wish is a bit complex to formalize. Still, we suggest robustness definitions that try to capture this idea.

Our definition of robust Bloom filter comes in several flavors, depending on whether the adversary aims to find a never-queried-before false-positive element or increase the false-positive rate; what the evaluation metric is, and depending on the information available to the adversary. We discuss each of these choices in turn.

*Adaptive vs. Non-adaptive Queries.* When discussing adaptivity, we refer to the settings where an adversary can choose the next query based on the response of the Bloom filter on previous ones.

Our wish that a robust Bloom filter behaves like a truly unpredictable biased coin can be hard to meet, even in the non-adaptive case. False-positive elements are not necessarily random independent events, e.g., if the universe is divided into pairs, and both the elements in each pair are either positive or negatives.

However, the problem is more serious when discussing the adaptive case. In the above example, an adversary can query one element in each pair and query the other only if the first one is positive, resulting in a higher false-positive rate.

Therefore, we consider adaptive settings when defining robust Bloom filters: the adversary can adaptively query the filter. Nevertheless, we also analyze the non-adaptive settings and, more precisely, the ability of an adversary to predict a false positive element in the case of non-adaptive queries. We analyze the non-adaptive settings to understand the significance of seeing the responses of the Bloom filter.

*One-Time Challenge vs. "Continuous" Challenge.* We consider both a one-time challenge and a continuous challenge; By one-time challenge, we refer to tests in which an adversary performs $t$ adaptive queries, and her goal is to find **one** never-queried-before false positive.

Some Bloom filter applications are sensitive to clusters of false positives, e.g. when Bloom filters are used to hold the content of a cache. An adversary that finds many false positives can cause unsuccessful cache access for almost every query, resulting in a Denial of Service (DoS) attack. Motivated by this, we also consider "continuous" tests, which examine the false-positive rate in a sequence of $t$ adaptive queries.

### 3.2 Robustness in All Shapes and Forms

We will describe five definitions of robustness for Bloom filters capturing various aspects of robustness. For each definition we outline a **test** for the Bloom filter. If the Bloom filter is resilient wrt the test against all adversaries, then we say that it is robust under the corresponding definition. In each of these tests, the adversary performs $t$ queries with a different challenge to achieve, which gives rise to a different type of robust Bloom filter. The difference between the definitions is the goal of the adversary and what she has access to, summarized in Table 1.

The conclusion of this investigation into the various notions of robustness is that the most desired notion is Bet-or-Pass.

**The Always-Bet (AB) Test** Our starting point is the definition of [23]: the adversary participates in an adaptive game $\mathsf{AdaptiveGame}_{\mathcal{A},t}(\lambda)$ and then outputs an element $x^*$ that was not queried before (and does not belong to $S$), which she believes is a false positive. The robustness is defined by the probability that the element is indeed a false positive.

*The AB Test* $\mathsf{ABTest}_{\mathcal{A},t}(\lambda)$

1. $\mathcal{A}$ participate in $\mathsf{AdaptiveGame}_{\mathcal{A},t}(\lambda)$ (Def. 2).
2. $\mathcal{A}$ outputs $x^*$.
3. The result of the test is 1 if $x^* \notin S \cup \{x_1, \ldots, x_t\}$ and $\mathbf{B}_2(M, x^*) = 1$, and 0 otherwise. If $\mathsf{ABTest}_{\mathcal{A},t}(\lambda) = 1$, we say that $\mathcal{A}$ made the Bloom filter fail.

| Test Name | Queries | Adversary Goal |
|---|---|---|
| Always-Bet | Adaptive | Find a never-queried-before false positive element |
| Bet-or-Pass | Adaptive | Bet on a never-queried-before false positive element or pass |
| Monotone | Adaptive | Find an event that happens more frequently (non-negligibly) than a truly random coin tosses |
| Expected Count | Adaptive | Find more than $\varepsilon \cdot t$ false positive in expectation |
| Semi-Adaptive Prediction | **Non**-Adaptive | Find a false positive among the queries chosen beforehand |

**Table 1.** A table comparing the settings and the adversary goal within different robustness definitions.

**Definition 7 (Always-Bet (AB) Test[9]).** *Let $\boldsymbol{B} = (\boldsymbol{B}_1, \boldsymbol{B}_2)$ be an $(n, \varepsilon)$-Bloom filter. We say that $\boldsymbol{B}$ is an $(n, t, \varepsilon)$-Always-Bet (AB) test resilient if for any probabilistic polynomial-time adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ there exists a negligible function* $\mathsf{negl}$ *such that:*

$$\Pr[\mathsf{ABTest}_{\mathcal{A},t}(\lambda) = 1] \leq \varepsilon + \mathsf{negl}(\lambda),$$

*where the probabilities are taken over the internal randomness of $\boldsymbol{B}$ and $\mathcal{A}$.*

**The Bet-or-Pass (BP) Test** The the adversary in the AB-test (Def. 7) **must** output a challenge element $x^*$. We suggest a definition that allows the adversary to pass; the adversary does not have to output an element. We define an adversary's *profit* : she gets rewarded if her output element is a false positive, while she is penalized if she outputs a true negative. She does not gain or lose any value when she chooses to pass. We use the expected profit to define the robustness: we want the expected profit to be 0. More formally, the adversary participate in an $\mathsf{AdaptiveGame}_{\mathcal{A},t}(\lambda)$. She outputs $(b, x^*)$ where $x^* \notin S \cup \{x_1, \ldots, x_t\}$ is the challenge and $b \in \{0, 1\}$ indicates whether she chooses to bet $(b = 1)$ or to pass $(b = 0)$. If she passes, $x^*$ is ignored and can be a random element.

*The BP Test* $\mathsf{BPTest}_{\mathcal{A},t}(\lambda)$

1. $\mathcal{A}$ participate in $\mathsf{AdaptiveGame}_{\mathcal{A},t}(\lambda)$ (Def. 2).
2. $\mathcal{A}$ outputs $(b, x^*)$.
3. $\mathcal{A}$'s profit $C_{\mathcal{A}}$ is defined as:

$$C_A = \begin{cases} \frac{1}{\varepsilon}, & \text{if } x^* \text{ is a false positive and } b = 1, \\ -\frac{1}{1-\varepsilon}, & \text{if } x^* \text{ is } \textbf{not} \text{ a false positive and } b = 1, \\ 0, & \text{if } b = 0. \end{cases}$$

[9] In [23] resilience to this test is simply referred to as adversarial resilient Bloom filter.

**Definition 8 (Bet-or-Pass (BP) Test).** *Let $\boldsymbol{B} = (\boldsymbol{B}_1, \boldsymbol{B}_2)$ be an $(n, \varepsilon)$-Bloom filter. We say that $\boldsymbol{B}$ is an $(n, t, \varepsilon)$-Bet-or-Pass (BP) test resilient if for every probabilistic polynomial-time adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ participating in $\mathsf{BPTest}_{\mathcal{A},t}(\lambda)$, there exits a negligible function* $\mathsf{negl}$ *such that:*

$$\mathbb{E}[C_{\mathcal{A}}] \leq \mathsf{negl}(\lambda).$$

*The expectation is taken over the internal randomness of $\boldsymbol{B}$ and $\mathcal{A}$.*

Note that the expected profit of an adversary outputting a random guess with probability at most $\varepsilon$ to be a false positive is at most 0:

$$\mathbb{E}[C_{\mathcal{A}}] = \underbrace{\Pr[x^* \text{ is FP } \wedge\ b = 1]}_{\leq \varepsilon} \cdot \frac{1}{\varepsilon} - \underbrace{\Pr[x^* \text{ is \textbf{not} FP } \wedge\ b = 1]}_{\geq 1-\varepsilon} \cdot \frac{1}{1-\varepsilon} \leq 0$$

The BP test allows an adversary to pass (although if she wants any chance to make the Bloom filter fail, the probability she passes must be noticeably far from 1). Adding the pass option suggests that the BP test is a stronger requirement and more robust notion than the AB test: consider a Bloom filter that behaves "nicely" in most cases except for a few bad cases. The probability of occurrence of the bad cases is negligible. Then this Bloom filter is AB test resilient. However, it is not BP test resilient. For the BP test, an adversary can become "active" (bets) only when she observes the occurrence of some bad cases. Given such bad cases, her success probability is not negligible.

We support this intuition by showing that BP test resilience implies AB test resilience, while the other direction does not necessarily hold. In addition, we consider another family of tests: **the monotone tests**, which resilience to them is implied by the BP test but not by the AB test.

One may note two differences between the AB and BP tests: the adversary must always provide a candidate false positive in the AB test, while it is optional in the BP test. In addition, they differ in the robustness metric: the probability of outputting a false positive vs. the expected profit. However, for the adversaries that always output an element with $b = 1$ we show that the robustness metric is equivalent (in Claim 3.2 below), meaning that allowing the adversary the option to pass is the actual difference between the two.

Let $\mathcal{A}$ be an adversary performing $\mathsf{ABTest}_{\mathcal{A},t}(\lambda)$ (Def. 7). We can think of it as an adversary performing $\mathsf{BPTest}_{\mathcal{A},t}(\lambda)$ (Def. 8) with $b = 1$ always. Therefore her expected profit is:

$$\mathbb{E}[C_{\mathcal{A}}] = \Pr[x^* \text{ is FP}] \cdot \frac{1}{\varepsilon} - \Pr[x^* \text{ is \textbf{not} FP}] \cdot \frac{1}{1-\varepsilon}$$

*Claim.* Let $\mathcal{A}$ be an adversary in $\mathsf{BPTest}_{\mathcal{A},t}(\lambda)$ game that always sets $b = 1$. Then there exists a negligible function $\mathsf{negl}_1$ such that $\Pr[x^* \text{ is FP}] \leq \varepsilon + \mathsf{negl}_1(\lambda)$ iff there exists a negligible function $\mathsf{negl}_2$ such that $\mathbb{E}[C_{\mathcal{A}}] \leq 0 + \mathsf{negl}_2(\lambda)$.

*Proof.* Let $\mathcal{A}$ be an adversary in the $\mathsf{BPTest}_{\mathcal{A},t}(\lambda)$ game that always sets $b = 1$. Assume that there exists a negligible function $\mathsf{negl}_1$ such that

$$\Pr[x^* \text{ is FP}] \leq \varepsilon + \mathsf{negl}_1(\lambda).$$

Then,

$$\begin{aligned}\mathbb{E}[C_{\mathcal{A}}] &= \Pr[x^* \text{ is FP}] \cdot \frac{1}{\varepsilon} - \Pr[x^* \text{ is } \textbf{not} \text{ FP}] \cdot \frac{1}{1-\varepsilon} \\ &\leq (\varepsilon + \mathsf{negl}_1(\lambda)) \cdot \frac{1}{\varepsilon(1-\varepsilon)} - \frac{1}{1-\varepsilon} \\ &= \frac{\mathsf{negl}_1(\lambda)}{\varepsilon(1-\varepsilon)} \\ &\leq \mathsf{negl}_2(\lambda),\end{aligned}$$

for some negligible function $\mathsf{negl}_2$. The last inequality follows from Inequality (1). Now, assume that there exists a negligible function $\mathsf{negl}_2$ such that

$$\mathbb{E}[C_{\mathcal{A}}] \leq \mathsf{negl}_2(\lambda).$$

Therefore,

$$\begin{aligned}\mathbb{E}[C_{\mathcal{A}}] = \Pr[x^* \text{ is FP}] \cdot \frac{1}{\varepsilon} - \Pr[x^* \text{ is } \textbf{not} \text{ FP}] \cdot \frac{1}{1-\varepsilon} &\leq \mathsf{negl}_2(\lambda) \\ \Pr[x^* \text{ is FP}] \cdot \frac{1}{\varepsilon(1-\varepsilon)} &\leq \frac{1}{1-\varepsilon} + \mathsf{negl}_2(\lambda) \\ \boxed{\Pr[x^* \text{ is FP}] \leq \varepsilon + \underbrace{\varepsilon(1-\varepsilon)}_{\leq 0.5} \cdot \mathsf{negl}_2(\lambda) \leq \varepsilon + \mathsf{negl}_1(\lambda)}&,\end{aligned}$$

for some negligible function $\mathsf{negl}_1$, as desired.

**Monotone Efficient Tests** Recall our desire (“wishful thinking") that a “robust” Bloom filter should behave like a truly unpredictable biased coin; It is not clear for both the AB and BP tests whether they satisfy this wish. To treat it more formally we consider monotone tests. A monotone test consists of $\mathsf{AdaptiveGame}_{\mathcal{A},t}(\lambda)$ as before, but now we are not interested in any specific output element. The test examines the response of the Bloom filter on $t$ adaptive queries performed by an adversary $\mathcal{A}$. Continuing the idea of biased coin tosses, we would like to think of false positives as random independent events with a probability smaller or equal to $\varepsilon$. We compare the Bloom filter response on a sequence of $t$ adaptive queries to a sequence of independent biased bits of length $t$ with bias $\varepsilon$ (probability of 1 is $\varepsilon$).

In this test, we consider monotone functions. Informally, a monotone function is a function that can only increase when we flip a 0 in the input string to 1:

**Definition 9 (Monotone Function).** *Let $t \in \mathbb{N}$. We say that a function $f\colon \{0,1\}^t \to \{0,1\}$ is* monotone *if for every pair of neighboring strings $x, x' \in \{0,1\}^t$ that are equal in all locations except in one index $1 \leq i \leq t$, i.e. $x_j = x'_j$ for all $j \neq i$ and $x_i = 0$ and $x'_i = 1$, we have that $f(x) = 1$ implies that $f(x') = 1$.*

The probability of a false positive can be less than $\varepsilon$, though this is hardly damaging. We are interested in clusters of false positives. This is what the **monotone** property aims to model.

Let $f\colon \{0,1\}^t \to \{0,1\}$ be a monotone function and $D_f$ a polynomial time algorithm computing $f$. We consider distinguishers of the form $D_f$.

We now present the formal definition. The fundamental realization is that a robust Bloom filter should be resilient to *all (efficient) monotone tests.* That is, for any efficient monotone test (or distinguisher) $D$, the probability that $D$ returns 1, when given the Bloom filter responses on a sequence of adaptively selected queries, should be close (from below) to the probability that $D$ returns 1 when given an independent biased sequence of the same length and with bias $\varepsilon$.

**Definition 10 (Monotone Test Resilient).** *Let $\boldsymbol{B} = (\boldsymbol{B}_1, \boldsymbol{B}_2)$ be an $(n,\varepsilon)$-Bloom filter. We say that $\boldsymbol{B}$ is $(n,t,\varepsilon)$-monotone test resilient if for every monotone probabilistic polynomial-time algorithm (distinguisher) $D\colon \{0,1\}^t \to \{0,1\}$ and every probabilistic polynomial-time adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ participating in an $\mathsf{AdaptiveGame}_{\mathcal{A},t}(\lambda)$ there exits a negligible function $\mathsf{negl}$ such that:*

$$\Pr_{S \in G_{\mathcal{A}}}[D(S) = 1] - \Pr_{S_\varepsilon \in B_\varepsilon}[D(S_\varepsilon) = 1] \leq \mathsf{negl}(\lambda)$$

*where $G_{\mathcal{A}}$ is the distribution of the Bloom filter outcomes on $\mathcal{A}$'s $t$ queries and $B_\varepsilon$ is a distribution of independent biased sequence of length $t$ with bias $\varepsilon$.*

Note the similarity and differences with the notion of cryptographic pseudorandomness (see Goldreich [13]). In our setting we consider only *monotone* polynomial-time tests (whereas in the notion of cryptographic pseudorandomness *all* polynomial-time tests are considered) and we look at the difference between the probabilities without an absolute value (whereas there the absolute value should be negligible).

We give examples for relevant monotone tests. The first one is the **FP's count distinguisher**, denoted by $D_w$ for some $w < t$. Let $s \in \{0,1\}^t$. We define,

$$D_w(s) = \begin{cases} 1, & \text{if } \#1 \text{ in } s \text{ is greater than } w, \\ 0, & \text{otherwise.} \end{cases}$$

$D_w$ outputs 1 iff the number of ones (equivalently, false-positive elements) is greater than $w$. $D$ is monotone. Another example is a **Cluster distinguisher** that outputs 1 iff the sequence contains $w$ consecutive ones. Indeed, this definition captures if a cluster of false positives is found in a Bloom filter response since the probability of finding a cluster of "1" is negligible in a bias coin tosses.

**The Expected Count Test** We use $\mathsf{AdaptiveGame}_{\mathcal{A},t}(\lambda)$. Similar to the monotone test, we are not interested in any output. Inspired by Clayton et al. [8], and as a special case of the monotone test, we look at the number of false positive elements that an adversary finds during her $t$ adaptive queries. Let $\mathcal{A}$ be an adversary participating in $\mathsf{AdaptiveGame}_{\mathcal{A},t}(\lambda)$ (Def. 2). Let $Q = \{x_1, \ldots, x_t\}$ be the queries performed by $\mathcal{A}$. Denote the number of false positive queries by $\#\mathsf{FP}_t \coloneqq |\{x_i \mid \mathbf{B}_2(M, x_i) = 1 \text{ and } x_i \in Q \setminus S\}|$.[10] We want to upper bound the expected number of false positives queries. Formally,

**Definition 11 (Expected Count Test).** *Let $\boldsymbol{B} = (\boldsymbol{B}_1, \boldsymbol{B}_2)$ be an $(n, \varepsilon)$-Bloom filter. We say that $\boldsymbol{B}$ is $(n, t, \varepsilon)$-expected count test resilient if for any probabilistic polynomial-time adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ participating in $\mathsf{AdaptiveGame}_{\mathcal{A},t}(\lambda)$ there exists a negligible function $\mathsf{negl}$ such that:*

$$\mathbb{E}[\#\mathsf{FP}_t] \leq \varepsilon \cdot t + \mathsf{negl}(\lambda),$$

*where the expectation is taken over the internal randomness of $\boldsymbol{B}$ and $\mathcal{A}$.*

**The Semi-Adaptive Prediction Test** Finally, we define a semi-adaptive test: the adversary chooses the queries in advance (non-adaptively) and needs to find a false positive element using oracle access to the Bloom filter (adaptively). This test allows us to evaluate the "power" of adaptive queries.

We formalize this by defining a game, $\mathsf{SemiAdaptiveGame}_{\mathcal{A},t}(\lambda)$. This is done in a fashion similar to $\mathsf{AdaptiveGame}_{\mathcal{A},t}(\lambda)$: we consider a polynomial-time adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ that consists of two parts: $\mathcal{A}_1$ chooses the set $S$ and commits to $t$ distinct queries $x_1, \ldots, x_t$, and $\mathcal{A}_2$ gets as input the set $S$, the queries $x_1, \ldots, x_t$ and oracle access to the query algorithm (initialized with $M$). $\mathcal{A}_2$ aims to find a false positive element among the $t$ queries without querying this element explicitly.

*The semi-adaptive game* $\mathsf{SemiAdaptiveGame}_{\mathcal{A},t}(\lambda)$*:*

1. The adversary $\mathcal{A}_1$ is given input $1^{\lambda + n \log u}$ and outputs a set $S \subset U$ of size $n$ and $t$ distinct queries $x_1, \ldots, x_t$ .[11]
2. $\mathbf{B}_1$ is given input $\left(1^{\lambda + n \log u}, S\right)$ and builds a representation $M$.
3. The adversary $\mathcal{A}_2$ is given input $\left(1^{\lambda + n \log u}, S, (x_1, \ldots, x_t)\right)$ and oracle access to $\mathbf{B}_2(M, \cdot)$. For $i \in [t]$:
   (a) $\mathcal{A}_2$ chooses one of the following: bet on $x_i$ to be a false positive **or** query $\mathbf{B}_2(M, x_i)$. If $\mathcal{A}_2$ choose to bet, then $x^* \leftarrow x_i$ and the game is stopped. Else, she continues.
4. $x^* \leftarrow x_t$
5. The result of the game is 1 if $x^* \notin S$ and $\mathbf{B}_2(M, x^*) = 1$, and 0 otherwise. If $\mathsf{SemiAdaptiveGame}_{\mathcal{A},t}(\lambda) = 1$, we say that $\mathcal{A}$ made the Bloom filter fail.

**Definition 12 (Semi-Adaptive Prediction Resilient).** *Let $\boldsymbol{B} = (\boldsymbol{B}_1, \boldsymbol{B}_2)$ be an $(n, \varepsilon)$-Bloom filter. We say that $\boldsymbol{B}$ is an $(n, t, \varepsilon)$-semi adaptive prediction*

[10] Note that we count the number of false positives without duplicates to not over-credit the adversary.

[11] For convenience, we treat the set of queries as an ordered set. The order can be determined by the adversary when she queries $\mathbf{B}_2$.

*resilient Bloom filter if for every probabilistic polynomial-time adversary* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ *there exits a negligible function* $\mathsf{negl}$ *such that:*

$$\Pr\left[\mathsf{SemiAdaptiveGame}_{\mathcal{A},t}(\lambda) = 1\right] \leq \varepsilon + \mathsf{negl}(\lambda).$$

Note that we allow the adversary to repeat queries in all the mentioned above definitions, though repeated queries are not counted. We are only interested in its ability to find "fresh" false positive elements, as opposed to [2] where the Bloom filter false-positive rate has to be at most $\epsilon$ even if an adversary repeat the same query $t$ times. The latter guarantee forces the Bloom filter to update its internal state after each query, while in our case, it is unnecessary but allowed.

## 4 Relationships Between the Various Notions of Robustness

In Section 3 we defined five different robustness tests. This section shows the relationships between them: which test gives us the most robust Bloom filter and which are the weakest tests. As we shall see, the most desirable notion for a robust Bloom filter is that of Bet-or-Pass. This notion satisfies our desired three requirements: first, it is sufficient, meaning it satisfies the security requirements. Second, it is not too strong: we present a construction of a Bloom filter satisfying the Bet-or-Pass definition. Finally, it is easy to use: it is formalized as a simple test for a Bloom filter.

### 4.1 Implications

We begin by showing which definition implies which (see Fig. 2). All the implications are true in the strong way; that is if $\mathsf{Test}_1$ implies $\mathsf{Test}_2$ then a Bloom filter is strongly $\mathsf{Test}_1$ resilient is also strongly $\mathsf{Test}_2$ resilient. We present our results considering a polynomial-time adversary; however, they also apply against unbounded adversaries if $t$ is known in advance.

$$\begin{tikzcd}
\textbf{BP Test Resilience} \arrow[r, Rightarrow, "2"] \arrow[d, Rightarrow, "1"] & \textbf{Monotone Test Resilience} \arrow[dd, Rightarrow, "5"] \\
\textbf{AB Test Resilience} \arrow[d, Rightarrow, "4"] \arrow[dr, Rightarrow, "3"] & \\
\textbf{Semi-Adaptive Prediction Resilience} & \textbf{Expected Count Test Resilience}
\end{tikzcd}$$

**Fig. 2.** Tests' Implications

### The BP Test

**Theorem 1.** *Let $0 < \varepsilon < 1$ and $n \in \mathbb{N}$. Let $\boldsymbol{B}$ be an $(n, \varepsilon)$-strongly BP test resilient Bloom filter. Then $\boldsymbol{B}$ is an $(n, \varepsilon)$-strongly AB test resilient Bloom filter.*

*Proof.* Appears in the proof of Claim 3.2.

**Theorem 2.** *Let $0 < \varepsilon < 1$ and $n \in \mathbb{N}$. Let $\boldsymbol{B}$ be an $(n, \varepsilon)$-strongly BP test resilient Bloom filter. Then $\boldsymbol{B}$ is also $(n, \varepsilon)$-strongly monotone test resilient. .*

*Proof.* Suppose, towards contradiction, that $\mathbf{B}$ is not an $(n, \varepsilon)$-strongly monotone test resilient Bloom filter. There exists a monotone probabilistic polynomial-time test $D$ and probabilistic polynomial-time adversary $\mathcal{A}$ performing $t \leq poly(\lambda, n)$ queries such that $D$ distinguish $G_{\mathcal{A}}$, the vector indicating whether a false positive occurred or not, from the biased independent sequence $B_\varepsilon$; that is, for some polynomial $p$ and infinitely many $\lambda$'s,

$$\Pr_{S \in G_{\mathcal{A}}}[D(S) = 1] - \Pr_{S_\varepsilon \in B_\varepsilon}[D(S_\varepsilon) = 1] \geq \frac{1}{p(\lambda)} \tag{2}$$

For each $\lambda$ satisfying Eq.(2), recall that $t \leq q(\lambda)$ for some polynomial $q$. We define $t+1$ hybrids. The $i$-th hybrid $(i = 0, 1, \ldots, t)$, denoted $H_\lambda^i$, consists of the $i$-bit long prefix of $G_{\mathcal{A}}$ followed by the $(t-i)$-bit long suffix of $B_\varepsilon$.

*Claim.* There exists $i^* \in \{1, \ldots, t\}$ such that

$$\Pr_{S \in H_\lambda^{i^*}}[D(S) = 1] - \Pr_{S \in H_\lambda^{i^*-1}}[D(S) = 1] \geq \frac{1}{p(\lambda) \cdot t}. \tag{3}$$

*Proof.* The proof is immediate by Eq. (2), the pigeonhole principle and the definition of the hybrids. In particular, we use the fact that $H_\lambda^t = G_{\mathcal{A}}$ and $H_\lambda^0 = B_\varepsilon$.

We now define an adversary $\mathcal{A}_{BP}$ for the BP test. For simplicity of the analysis, we assume that $\mathcal{A}_{BP}$ knows $i^*$. The idea is that monotonicity implies that an adversary can know when to bet. $\mathcal{A}_{BP}$ produce a $(i^* - 1)$-bit long prefix using $\mathcal{A}$'s queries and a $(t - i^* - 1)$-bit long suffix that contains random biased bits. Then, she gives the distinguisher the concatenated sequence twice: one time when there is 0 in the $i^*$-th index and one time where there is 1. If the distinguisher is sensitive to this change, then $\mathcal{A}_{BP}$ chooses to bet on $x_{i^*}$. Otherwise, she passes.

*Adversary $\mathcal{A}_{BP}$*

1. Set $i = i^*$
2. Run $\mathcal{A}$ for $i-1$ queries $x_1, \ldots, x_{i-1}$. For each $j \in [i-1]$ let $y_j = \mathbf{B}(S, x_j)$.
3. Select $r_{i+1}, \ldots, r_t$ independently with bias $\varepsilon$ in $\{0, 1\}$ $(\Pr[r_j = 1] = \varepsilon)$.
4. If $D(y_1, \ldots, y_{i-1}, 1, r_{i+1}, \ldots, r_t) \neq D(y_1, \ldots, y_{i-1}, 0, r_{i+1}, \ldots, r_t)$, then bet $b = 1$ and output $\mathcal{A}$'s $i$th query $x_i$.
5. Else, pass: $b = 0$ (meaning, the adversary does not bet in any round)

To analyze the success of $\mathcal{A}_{BP}$ define two sets of sequences from $\{0,1\}^{t-1}$. The first one is $\mathsf{BET}_i$ which is defined as

$$\{y_1,\ldots,y_{i-1},r_{i+1},\ldots,r_t|D(y_1,\ldots,y_{i-1},1,r_{i+1},\ldots,r_t)\neq D(y_1,\ldots,y_{i-1},0,r_{i+1},\ldots,r_t)\}$$

and the second one is $\mathsf{ONE}_i$ which is defined as

$$\{y_1,\ldots,y_{i-1},r_{i+1},\ldots,r_t|D(y_1,\ldots,y_{i-1},1,r_{i+1},\ldots,r_t)= D(y_1,\ldots,y_{i-1},0,r_{i+1},\ldots,r_t)=1\}.$$

By the monotonicity of $D$ we have that if

$$D(y_1,\ldots,y_{i-1},1,r_{i+1},\ldots,r_t)\neq D(y_1,\ldots,y_{i-1},0,r_{i+1},\ldots,r_t),$$

then $D(y_1,\ldots,y_{i-1},1,r_{i+1},\ldots,r_t)=1$ and $D(y_1,\ldots,y_{i-1},0,r_{i+1},\ldots,r_t)=0$. Using this notation we have:

$$\begin{aligned}\Pr_{S\in H_\lambda^{i^*}}[D(S)=1] &= \Pr[y_1,\ldots,y_{i^*-1},r_{i^*+1},\ldots,r_t\in\mathsf{BET}_{i^*}\wedge x_{i^*}\text{ is FP}]\\ &\quad+\Pr[y_1,\ldots,y_{i^*-1},r_{i^*+1},\ldots,r_t\in\mathsf{ONE}_{i^*}]\end{aligned}$$

and,

$$\begin{aligned}\Pr_{S\in H_\lambda^{i^*-1}}[D(S)=1] &= \varepsilon\cdot\Pr[y_1,\ldots,y_{i^*-1},r_{i^*+1},\ldots,r_t\in\mathsf{BET}_{i^*}]\\ &\quad+\Pr[y_1,\ldots,y_{i^*-1},r_{i^*+1},\ldots,r_t\in\mathsf{ONE}_{i^*}],\end{aligned}$$

where the probabilities are over the internal randomness of $\mathbf{B}$ and $\mathcal{A}$ and the biased coin flips. Combining the above with Eq.(3) we get:

$$\begin{aligned}\frac{1}{p(\lambda)\cdot t} &\leq \Pr_{S\in H_\lambda^{i^*}}[D(S)=1]-\Pr_{S\in H_\lambda^{i^*-1}}[D(S)=1]\\ &=\Pr[y_1,\ldots,y_{i^*-1},r_{i^*+1},\ldots,r_t\in\mathsf{BET}_{i^*}\wedge x_{i^*}\text{ is FP}]\\ &-\varepsilon\cdot\Pr[y_1,\ldots,y_{i^*-1},r_{i^*+1},\ldots,r_t\in\mathsf{BET}_{i^*}].\end{aligned}$$

Hence we get that the probability that $\mathcal{A}_{BP}$ bets is non-negligible:

$$\Pr[b=1]=\Pr[y_1,\ldots,y_{i^*-1},r_{i^*+1},\ldots,r_t\in\mathsf{BET}_{i^*}]\geq\frac{1}{p(\lambda)\cdot t\cdot(1-\varepsilon)},$$

and the probability that $\mathcal{A}_{BP}$ outputs a false positive element, when she bets, is noticeably greater than $\varepsilon$:

$$\begin{aligned}\Pr[x_{i^*}\text{ is FP}\mid b=1] &= \Pr[x_{i^*}\text{ is FP}\mid y_1,\ldots,y_{i^*-1},r_{i^*+1},\ldots,r_t\in\mathsf{BET}_{i^*}]\\ &=\frac{\Pr[x_{i^*}\text{ is FP}\wedge y_1,\ldots,y_{i^*-1},r_{i^*+1},\ldots,r_t\in\mathsf{BET}_{i^*}]}{\Pr[y_1,\ldots,y_{i^*-1},r_{i^*+1},\ldots,r_t\in\mathsf{BET}_{i^*}]}\\ &\geq\frac{1}{p(\lambda)\cdot t}+\varepsilon.\end{aligned}$$

Therefore the expected profit of $\mathcal{A}_{BP}$ is noticeably greater then 0, as desired.

**The AB Test**

**Theorem 3.** *Let $0 < \varepsilon < 1$ and $n \in \mathbb{N}$. Let $\boldsymbol{B}$ be an $(n, \varepsilon)$-strongly AB test resilient Bloom filter. Then $\boldsymbol{B}$ is also $(n, \varepsilon)$-expected count test resilient.*

*Proof.* Let $0 < \varepsilon < 1$, $n \in \mathbb{N}$ and let $\mathbf{B}$ be an $(n, \varepsilon)$-strongly AB test resilient Bloom filter. Assume, for contradiction, that $\mathbf{B}$ is not an $(n, \varepsilon)$-strongly expected count test resilient. Meaning there exists a PPT adversary $\mathcal{A}$ that makes at most $t \leq poly(\lambda, n)$ queries, a polynomial $p(\cdot)$ such that for infinitely many $\lambda$'s

$$\mathbb{E}[\#\mathsf{FP}_t] \geq \varepsilon \cdot t + \frac{1}{p(\lambda)}.$$

For convenience we assume that the queries were distinct. We use $\mathcal{A}$ to build an adversary $\mathcal{A}'$ that causes $\mathbf{B}$ to fail in the $(n, t, \varepsilon)$-AB test.

*Adversary $\mathcal{A}'$*

1. Choose a random number $j \in [t]$.
2. Runs $\mathcal{A}$ for $j - 1$ queries using oracle access to $\mathbf{B}$.
3. Output $x_j$.

Observe that

$$\mathbb{E}[\#\mathsf{FP}_t] = \mathbb{E}\left[\sum_{i=1}^{t} \mathbb{1}_{\{x_i \text{ is FP}\}}\right] = \sum_{i=1}^{t} \Pr[x_i \text{ is FP}].$$

Then,

$$\Pr[\mathsf{ABTest}_{\mathcal{A}',t}(\lambda) = 1] = \frac{\sum_{i=1}^{t} \Pr[x_i \text{ is FP}]}{t} \geq \varepsilon + \frac{1}{p(\lambda)t} \geq \varepsilon + \frac{1}{q(\lambda)},$$

for some polynomial $q(\cdot)$, where in the last inequality we used the fact that $t \leq poly(\lambda, n)$.

**Theorem 4.** *Let $0 < \varepsilon < 1$ and $n \in \mathbb{N}$. Let $\boldsymbol{B}$ be an $(n, \varepsilon)$-strongly AB test resilient Bloom filter. Then $\boldsymbol{B}$ is an $(n, \varepsilon)$-strongly semi-adaptive prediction resilient.*

*Proof.* Immediate by definition: semi-adaptive prediction resilience is a special case of the AB test with non-adaptive queries.

**Monotone Tests**

**Theorem 5.** *Let $0 < \varepsilon < 1$ and $n \in \mathbb{N}$. Let $\boldsymbol{B}$ be an $(n, \varepsilon)$-strongly monotone test resilient Bloom filter. Then $\boldsymbol{B}$ is an $(n, \varepsilon)$-strongly expected count test resilient.*

*Proof.* Let $0 < \varepsilon < 1$, $n \in \mathbb{N}$ and let $\mathbf{B}$ be an $(n, \varepsilon)$-strongly monotone test resilient Bloom filter. Assume, for contradiction, that $\mathbf{B}$ is not an $(n, \varepsilon)$-strongly expected count test resilient. Meaning there exists a PPT adversary $\mathcal{A}$ that makes at most $t \leq poly(\lambda, n)$ queries , a polynomial $p(\cdot)$ such that for infinitely many $\lambda$'s

$$\mathbb{E}[\#\mathsf{FP}_t] \geq \varepsilon \cdot t + \frac{1}{p(\lambda)}.$$

Therefore, there must exist $1 \leq j \leq t$ s.t. for infinitely many $\lambda$'s

$$\Pr[x_j \text{ is FP}] \geq \varepsilon + \frac{1}{p(\lambda)t} \geq \varepsilon + \frac{1}{q(\lambda)}.$$

where in the right inequality we used the fact that $t \leq poly(\lambda, n)$. For every $j \in [t]$, we define a monotone test,

$$D_j = \begin{cases} 1, & \text{if the } j\text{-th index in the sequence is } 1, \\ 0, & \text{else.} \end{cases}$$

We show that $\mathbf{B}$ fails the test $D_j$, meaning it is not an $(n, \varepsilon)$-strongly monotone test resilient Bloom filter. Indeed,

$$\Pr_{S \in G_{\mathcal{A}}}[D_j(S) = 1] - \Pr_{S_\varepsilon \in B_\varepsilon}[D_j(S_\varepsilon) = 1] \geq \varepsilon + \frac{1}{q(\lambda)} - \varepsilon = \frac{1}{q(\lambda)},$$

as desired.

## 4.2 Separations

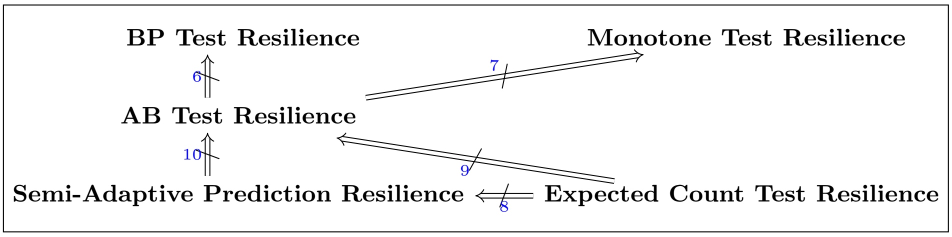

**Fig. 3.** Tests' Separations

We now show the separations between the various notions, as described in Fig. 3; that is, to show a separation between $\mathsf{Test}_1$ and $\mathsf{Test}_2$ we present a construction of a Bloom filter that is strongly $\mathsf{Test}_1$ resilient but is not strongly $\mathsf{Test}_2$ resilient.

**AB Test**

**Theorem 6.** *Let $0 < \varepsilon < 1$ and $n \in \mathbb{N}$, then for any $0 < \delta < 1$ and for large enough $u$:*

1. *Assuming the existence of one-way functions, then there exists a non-trivial Bloom filter $\boldsymbol{B}$ that is an $(n, \varepsilon)$-strongly AB test resilient and is not an $(n, \delta)$-strongly BP test resilient.*
2. *If $t$ is known in advance, then there exists a non-trivial Bloom filter $\boldsymbol{B}$ that is an $(n, t, \varepsilon)$-AB test resilient and is not an $(n, t, \delta)$-BP test resilient.*

*Proof.* Let $0 < \varepsilon < 1$, $0 < \varepsilon_1 < \varepsilon$, $n \in \mathbb{N}$. First, assume that $t$ is unknown and unbounded. To have an $(n, \varepsilon_1)$-strongly AB test resilient Bloom filter, we need to use Naor and Yogev construction that uses one-way functions—using that, and we have a Bloom filter $\mathbf{B}$ that is an $(n, \varepsilon_1)$-strongly AB test resilient. Let $\varepsilon_2 = \frac{\varepsilon - \varepsilon_1}{1 - \varepsilon_1}$. We consider the following Bloom filter $\mathbf{B}'$:

$$\mathbf{B}'(S, \cdot) \equiv \begin{cases} 1, & \text{w.p. } \varepsilon_2, \\ \mathbf{B}(S, \cdot), & \text{w.p. } 1 - \varepsilon_2. \end{cases}$$

That is, in the setup phase $\mathbf{B}'$ flips a coin with bias $\varepsilon_2$ to decide whether it always answers 1 (regardless of the input) or always answers as $\mathbf{B}$. We use the notation "$\equiv$" to denote the result of that coin toss. Let $\varepsilon = \varepsilon_2 + (1 - \varepsilon_2) \cdot \varepsilon_1$. The probability of false positive in $\mathbf{B}'$ equals $\varepsilon$; meaning $\mathbf{B}'$ is an $(n, \varepsilon)$-Bloom filter. Moreover, for all $t \leq poly(\lambda, n)$, $\mathbf{B}'$ is resilient to the $(n, t, \varepsilon)$-AB test:

$$\begin{aligned} \Pr[\mathsf{ABTest}_{\mathcal{A},t}(\lambda) = 1] &= \Pr[x^* \text{ is } \textit{false positive}] \\ &= \Pr\big[\mathbf{B}'(S, \cdot) \equiv 1 \vee \mathbf{B}'(S, x^*) \equiv \mathbf{B}(S, x^*) = 1\big] \\ &\leq \varepsilon_2 + \varepsilon_1 \cdot (1 - \varepsilon_2) = \varepsilon, \end{aligned}$$

where in the inequality we used the fact that $\mathbf{B}$ is an $(n, \varepsilon_1)$-strongly AB test resilient. We show that $\mathbf{B}$ is not an $(n, \delta)$-strongly BP test resilient for any $0 < \delta < 1$. Let $t \leq poly(\lambda, n)$ and $0 < \delta < 1$. We describe an adversary $\mathcal{A}$ that causes $\mathbf{B}'$ to fail in the $(n, t, \delta)$-BP test. $\mathcal{A}$ queries random elements in $U \setminus S$ to check whether we are in the "all 1" case where all the elements are false positive. If all the queries are indeed false positives, then with high probability, we are in the "all 1" case and $\mathcal{A}$ bets on a random element. Otherwise, she passes. Formally,

*Adversary $\mathcal{A}$*

1. Choose a random set $S \subset U$ of size $n$.
2. For $i \in [t]$:
   (a) Query independent random elements $x_i \in U \setminus S$.
3. If for all $i \in [t]$: $\mathbf{B}'(S, x_i) = 1$ (i.e., all the queried elements are false positive), choose a random element $x^*$ and output $(b = 1, x^*)$ (**bet**).
4. Otherwise, set $b = 0$ (**pass**).

First, note that the probability that $\mathcal{A}$ bets is non-negligible:

$$\Pr[\mathcal{A} \text{ bets}] \geq \Pr\bigl[\mathcal{A} \text{ bets} \mid \mathbf{B}' \equiv 1\bigr] \cdot \Pr\bigl[\mathbf{B}' \equiv 1\bigr] \geq 1 \cdot \varepsilon_2.$$

Now, assume that $\mathcal{A}$ chooses to bet (i.e., all the queries in step 2 are false-positive elements). Consider the following three cases:

1. We are in the "all 1" case. In this case, the profit is $1/\delta$.
2. We are not in the "all 1" case, and the false positive rate is greater or equal to $\delta$. In this case, the profit is non-negative.
3. Otherwise, the profit is negative, but this happens with probability at most $\delta^t$ (which is a very small probability for sufficiently large $t$)[12].

Summing up the above cases, we get that the expected profit of $\mathcal{A}$ is noticeably greater than zero[13], meaning $\mathbf{B}$ is not an $(n, \delta)$-strongly BP test resilient.

If $t$ is known in advance, we use Naor and Yogev construction of Bloom filter that is resilient to the $(n, t, \varepsilon_1)$-AB test against unbounded adversaries. The above proof holds for this specific $t$.

**Theorem 7.** *Let $0 < \varepsilon < 1$ and $n \in \mathbb{N}$, then for any $0 < \delta < 1$ and for large enough $u$:*

1. *Assuming the existence of one-way functions, then there exists a non-trivial Bloom filter $\boldsymbol{B}$ that is an $(n, \varepsilon)$-strongly AB test resilient and is not an $(n, \delta)$-strongly monotone test resilient.*
2. *If $t$ is known in advance, then there exists a non-trivial Bloom filter $\boldsymbol{B}$ that is an $(n, t, \varepsilon)$-AB test resilient and is not an $(n, t, \delta)$-monotone test resilient.*

*Proof.* Let $0 < \varepsilon < 1$, $0 < \varepsilon_1 < \varepsilon$ and $n \in \mathbb{N}$. We use the Bloom filter $\mathbf{B}'$ defined in the proof of Theorem 6. We show that $\mathbf{B}'$ is not monotone test resilient for any $0 < \delta < 1$ by proving it fails the **FP's count distinguisher**, denoted by $D_w$ (for some $w < t$). Recall,

$$D_w = \begin{cases} 1, & \text{if } \#1 \text{ in the input sequence is greater than } w, \\ 0, & \text{otherwise.} \end{cases}$$

We define the adversary $\mathcal{A}$ as follow:

*Adversary $\mathcal{A}$*

1. Choose a random set $S \subset U$ of size $n$.
2. For $i \in [t]$:
   (a) Query independent random elements $x_i \in U \setminus S$.

[12] Any $(n, \varepsilon)$ Bloom filter is robust when the number of queries is small: if $t$ is much less than $1/\varepsilon$ then we do not expect the adversary to see any false positives, and hence we can consider the queries as chosen in advance. Therefore, if $t$ is not large enough, it is not interesting.

[13] We want $\delta^t < \varepsilon_2$ leading to $u > t > \log_\delta \varepsilon_2$.

Observe that with probability at least $\varepsilon_2$ $\mathcal{A}$ can achieve as many false positives as she wants (w.p. $\varepsilon_2$ all the queries are false positives). Meaning for any $t \leq poly(\lambda, n)$ we have

$$\Pr_{S \in G_{\mathcal{A}}}[D_{t-1}(S) = 1] = \Pr[\#\mathsf{FP}_t > t-1] \geq \varepsilon_2.$$

On the other hand, for any $0 < \delta < 1$:

$$\Pr_{S_\delta \in B_\delta}[D_{t-1}(S_\delta) = 1] = \delta^t$$

Therefore,

$$\Pr_{S \in G_{\mathcal{A}}}[D_{t-1}(S) = 1] - \Pr_{S_\delta \in B_\delta}[D_{t-1}(S_\delta) = 1] \geq \varepsilon_2 - \delta^t \geq \frac{1}{p(\lambda)},$$

for some polynomial $p(\cdot)$ and sufficiently large $t$.

### Expected Count Test

**Theorem 8.** *Let $0 < \varepsilon < 1$ and $n \in \mathbb{N}$, then for any $0 < \delta < 1$ and for large enough $u$: There exists a Bloom filter $\boldsymbol{B}$ that is an $(n, \varepsilon)$-strongly expected count test resilient, and is not an $(n, \delta)$-strongly semi-adaptive prediction resilient.*

*Proof.* Let $0 < \varepsilon < 1$, $n \in \mathbb{N}$ and a set $S$ of size $n$. Let $\varepsilon_1 \leq \frac{\varepsilon}{2-\varepsilon}$ s.t. $\frac{1}{\varepsilon_1} \in \mathbb{N}$. We partition the universe into disjoint blocks of size $b := \frac{1}{\varepsilon_1}$. Let $\mathbf{B}$ be a Bloom filter that stores the set $S$ explicitly (we can modify the construction to work for non-trivial Bloom filters as well). We add "synthetic" false-positive elements to $\mathbf{B}$ in the following way: each block has exactly one false positive element (resulting in a false positive probability of $\varepsilon_1$ for random queries). In order to determine which element is positive in each set, we use a pseudorandom function[14] $\mathsf{PRF}$. The $\mathsf{PRF}$ gets as input the block name and outputs the positive element in the block.

As we shall see, the resulting Bloom filter $\mathbf{B}$ is an $(n, \varepsilon)$-strongly expected count test resilient. We first claim that the best strategy for an adversary in order to increase the expected number of false positives is querying each block until she finds the false positive. Consider an adversary $\mathcal{A}$ following this strategy and focus on one block. The expected number of queried elements until finding the false positive is $\frac{b+1}{2}$. Assume $\mathcal{A}$ queries $t'$ blocks (where $t' \gg b$). The false positive rate in this sequence is (with high probability):

$$\frac{t'}{t' \cdot \frac{b+1}{2}} = \frac{2}{b+1} = \frac{2\varepsilon_1}{1+\varepsilon_1} = \varepsilon.$$

[14] A pseudorandom function ($\mathsf{PRF}$) is a keyed function $F$ such that $F_k$ (for key $k$ chosen uniformly at random) is indistinguishable from a truly random function given only oracle access to the function. See Goldreich [13].

Now, consider an adversary $\mathcal{A}'$ that uses a different strategy, i.e., she queries blocks and might move on to another block before finding the false positive. Let $\mathcal{A}''$ be an adversary that follows $\mathcal{A}'$'s strategy with a slight change: every time $\mathcal{A}'$ moves on to another block before finding the false-positive, $\mathcal{A}''$ continues querying this block until she finds the false positive. Intuitively, it is better to keep querying an "open" block since we are left with fewer elements. Formally, let us look at all the queries $\mathcal{A}''$ added when continuing querying a block. They are divided into blocks of size at most $b-1$. Hence, the false-positive rate in these added queries, similarly to the above computation, is at least $2\varepsilon_1 > \varepsilon$. Now, consider the rest of $\mathcal{A}'$'s queries. They either contain blocks that a false positive was found in them or blocks with no false positive. As shown above, the expected number of queried elements before finding a false positive is $\frac{b+1}{2}$. Assume that there are $t_1$ blocks that a false positive was found in them and $t_2$ blocks with no false positive. In each block, we query at least one element hence the false positive rate in these queries is at most

$$\frac{t_1}{t_1 \cdot \frac{b+1}{2} + t_2} < \frac{t_1}{t_1 \cdot \frac{b+1}{2}} = \varepsilon.$$

Since $2\varepsilon_1 > \varepsilon$, we get that $\mathcal{A}$ can only improve the expected number of false positives of $\mathcal{A}'$, as desired.

We conclude that the expected number of false positives in $t$ queries is at most $\varepsilon t$, as desired.

On the other hand, $\mathbf{B}$ is not an $(n, \varepsilon)$-strongly semi-adaptive prediction resilient. Let $\mathcal{A}$ be an adversary querying $p(\lambda)$ blocks, for some polynomial $p(\cdot)$; that is, she performs $t = p(\lambda) \cdot \frac{1}{\varepsilon_1}$ queries. She acts as follows: she queries each block separately. When she gets the response of the Bloom filter on an entire block except for one element and does not see any false positive, she bets on the remaining element. Therefore,

$$\Pr\left[\mathsf{SemiAdaptiveGame}_{\mathcal{A},t}(\lambda) = 1\right] = 1 - (1-\varepsilon_1)^t \geq 1 - \frac{1}{q(\lambda)} \geq \delta + \frac{1}{s(\lambda)},$$

for any $0 < \delta < 1$, sufficiently large $\lambda$ and some polynomials $q(\cdot), s(\cdot)$.

**Theorem 9.** *Let $0 < \varepsilon < 1$ and $n \in \mathbb{N}$, then for any $0 < \delta < 1$ and for large enough $u$: There exists a Bloom filter $\boldsymbol{B}$ that is an $(n, \varepsilon)$-strongly expected count test resilient , and is not an $(n, \delta)$-strongly AB test resilient.*

*Proof.* Follows from the proof of Theorem 8.

### Semi-adaptive Prediction Resilient

**Theorem 10.** *Let $0 < \varepsilon < 1$ and $n \in \mathbb{N}$, then for any $0 < \delta < 1$ and for large enough $u$:*

1. *Assuming the existence of one-way functions, there exists a non-trivial Bloom filter $\boldsymbol{B}$ that is an $(n, \varepsilon)$-strongly semi-adaptive prediction resilient and is not an $(n, \delta)$-strongly AB test resilient.*

*2. If $t$ is known in advance and sufficiently large, there exists a non-trivial Bloom filter $\boldsymbol{B}$ that is an $(n, t, \varepsilon)$-semi-adaptive prediction resilient and is not an $(n, t, \delta)$-AB test resilient.*

The proof appears in the full version and is omitted due to space limitations.

### 4.3 Conclusions

We showed that if a Bloom filter is resilient to the BP test, it is resilient to all monotone tests. At the same time, this does not necessarily hold for a Bloom filter that is resilient to the AB test, demonstrating that the AB test can miss "bad" events such as clusters of false positives. We also proved that the BP test implies the AB test. Altogether we highlight the notion of *Bet-or-Pass* as capturing the desired properties of a robust Bloom filter.

## 5 Computational Assumptions and One-way Functions

### 5.1 Constructions of BP Resilient Filters using One-way Functions

What we know so far:

$$\begin{tikzcd}\text{BP Test Resilience} \arrow[r, Rightarrow, shift left, "\text{We showed}"] & \text{AB Test Resilience} \arrow[l, Rightarrow, shift left, "/" marking] \arrow[r, Rightarrow, shift left, "{[23]}"] & \text{OWF} \arrow[l, Rightarrow, shift left]\end{tikzcd}$$

The black arrows are existential equivalence and the blue arrows are definition implication (with the same parameters) or separation. Therefore, if one-way functions do not exist, any non-trivial Bloom filter fails the BP test. We show that the existence of one-way functions also implies BP test resilient Bloom filters. For that, we show a construction of a Bloom filter that is strongly BP test resilient using one-way functions.

*Pseudorandom Functions.* A pseudorandom function (PRF) is an efficiently computable, keyed function $F$ that is indistinguishable from a truly random function (given only oracle access to the function). A pseudorandom permutation (PRP) is a pseudorandom function such that $F$ is a permutation and can be both efficiently computable and efficiently invertible.

We can construct pseudorandom function from any (length-doubling) pseudorandom generators ([14]), which in turn can be based on one-way functions. In addition, we can obtain a pseudorandom permutations from pseudorandom functions (i.e., using Luby-Rackoff construction [19], [22]).

*Constructing BP Test Resilient Bloom Filters.* Our starting point is the transformation presented by Naor and Yogev. Assuming the existence of one-way functions they showed that any Bloom filter could be efficiently transformed into an $(n, \varepsilon)$-strongly AB test resilient Bloom filter using approximately the same amount of memory. The idea is simple: adding a layer of a pseudorandom permutation. That is, on input $x$, we compute a pseudorandom permutation of

$x$ and send it to the original Bloom filter. The main idea is that we make the queries look random by applying a pseudorandom permutation. Therefore an adversary has no significant advantage in choosing the queries adaptively. Note that the correctness properties remain when using the permutation. We ask if the above transformation also yields an $(n, \varepsilon)$-strongly BP test resilient Bloom filter. However, unlike the AB test, the BP test allows an adversary to pass. This gives rise to two potential attacks:

1. Assume that with some non-negligible probability, all the elements in the universe (excluding the set $S$) are false positives (e.g., the Bloom filter presented in separation 6). Applying a pseudorandom permutation, in that case, will make no difference; the adversary presented in separation 6 can still make this Bloom filter to fail the BP test even after adding the PRP layer.
2. The universe is of polynomial size, i.e., $|u| = poly(\lambda)$ and there exists an attacker that knows the exact number of false positives in the universe (this is not an unreasonable property of some constructions). In this case, the attack includes the adversary querying the entire universe $U$ except for one element $x^*$. Based on the number of false positives she has seen so far, she knows with high probability if $x^*$ is a false positive or not and chooses to bet or pass accordingly.

Therefore, we cannot use the transformation of Naor and Yogev when constructing a BP test resilient Bloom filter.[15]

Apart from the above mentioned transformation, Naor and Yogev presented a construction of a Bloom filter $\mathbf{B}$ that is an $(n, t, \varepsilon)$-AB test resilient against an *unbounded adversary and a given number $t$ of queries*, for any $n, t \in \mathbb{N}$ and $0 < \varepsilon < 1/2$. As we shall see, this construction is actually also good for the BP-test (when the adversary is limited to $t$ queries), i.e. $\mathbf{B}$ is also an $(n, t, \varepsilon)$-BP test resilient for any $n, t \in \mathbb{N}$ and $0 < \varepsilon < 1/2$.

We use this construction with a slight change to yield a Bloom filter $\mathbf{B}'$ against a *computationally bounded adversary when $t$ is not necessarily known and can be unbounded.*

We start by presenting Naor and Yogev's construction (which in turn builds on Carter et al. [7]). They suggested to use a Cuckoo Hashing implementation of dictionary ([26], [25]) to store the set. Roughly speaking, Cuckoo Hashing consists of two tables $T_1$ and $T_2$ and two hash functions $h_1$ and $h_2$. Each element $x$ is stored in either $T_1[h_1(x)]$ or $T_2[h_2(x)]$. Instead of storing $x$ in those locations, the value of an unpredictable (in the sense described below) function $g$ at point $x$ (i.e. $g(x)$) is stored at either $T_1[h_1(x)]$ or $T_2[h_2(x)]$ (see Fig. 4). When doing a lookup of $x$ the values stored in $T_1[h_1(x)]$ and $T_2[h_2(x)]$ are retrieved and compared to $g(x)$ and a yes is returned iff at least one of them is equal to $g(x)$. The range of $g$ should be about $\frac{2}{\varepsilon}$.

[15] It may be possible to modify the transformation to yield a BP test resilient Bloom filter; for instance, we can test if we are in case 1 and reselect the random bits or combat case 2 by adding as noise false positives. We will explore it in future work.

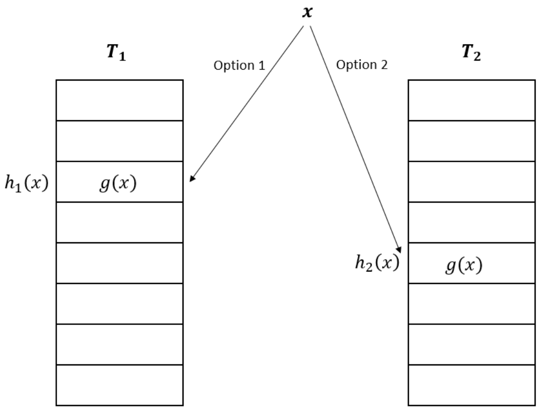

**Fig. 4.** Bloom Filter via Cuckoo Hashing

For the function $g\colon U \mapsto V$, they used a very high independence function. More formally, they used a family $G$ of hash functions satisfying that on any set of $k$ inputs, it behaves like a truly random function with high probability (based on the work of [24], and [10]). Note that the guarantee of the function still holds even when the set of queries is chosen adaptively, as shown by Berman et al. [3]. To reduce the memory size further, they use the family $G$ slightly differently. Let $\ell = O\big(\log \frac{1}{\varepsilon}\big)$, and set $k = O\big(\frac{t}{\ell}\big)$. They chose a family $G$ of functions $g_i$ that outputs a single bit (i.e., $V = \{0, 1\}$) and defined $g$ to be the concatenation of $\ell$ independent $g_i$ functions. Given a query $x$, they compare $g(x)$ to the appropriate entries, bit by bit. If the first two bits are equal, they continue to the next bit in a cyclic order. Consider an adversary performing $t$ queries. Naor and Yogev showed that even though the adversary performs $t$ queries, each of the $\ell$ different functions $g_i$ takes part in at most $O(t/\ell) = k$ queries (with high probability). Hence, each function $g_i$ still "looks" random on the queried elements. Therefore, we get a Bloom filter $\mathbf{B}$ that is AB test resilient for $t$ queries and uses $O\big(n \log \frac{1}{\varepsilon} + t\big)$ bits of memory.

The security of the scheme is based on the randomness properties of $g$. Even if all the values in the tables that have ever been used are known to the adversary (including the functions $h_1$ and $h_2$), the value of $g(x)$ is unknown and is uniform in its range. Therefore the probability that it is equal to the value stored in $T_1[h_1(x)]$ or $T_2[h_2(x)]$ is at most $2 \cdot (1/2)^\ell \leq \varepsilon/2$. Transforming from exact $k$-wise independence to almost $k$-wise independence adds an error probability of $\varepsilon/2$. Thus, the probability that $x$ is a false positive is at most $\varepsilon$. This proves that the construction is AB test resilient.

But this also means that there is no hint that a success is coming, i.e. that for the queried $x$ the value $g(x)$ is going to be equal to the values stored in locations $h_1(x)$ and $h_2(x)$. The only possible problem could be that more than $O(t/\ell) = k$ queries involve some $g_i$, but this happens with probability exponentially small in $k$. So we conclude that we can use this for the BP-test as well. Therefore, we get the following corollary:

**Corollary 1.** *For any $n, t \in \mathbb{N}$, universe of size $u \in \mathbb{N}$ and $0 < \varepsilon < 1/2$ there exists an $(n, t, \varepsilon)$-BP test resilient Bloom filter (against unbounded adversaries and $t$ known in advance) that uses $O\big(n \log \frac{1}{\varepsilon} + t\big)$ bits of memory. In fact, let $\boldsymbol{B}$ be a Bloom filter as described above. Then for any constant $0 < \varepsilon < 1/2$, $\boldsymbol{B}$ is an $(n, t, \varepsilon)$-BP test resilient Bloom filter against unbounded adversaries, that uses $m$ bits of memory where $m = O\big(n \log \frac{1}{\varepsilon} + t\big)$.*

Note that $t$ needs to be known in advance in this construction (to set $k$ and choose appropriate $G$).

To get a Bloom filter $\mathbf{B}'$ that is an $(n, \varepsilon)$-strongly BP test resilient, we will modify this construction a bit. The idea is simple: for the function $g$, we use a family of pseudorandom functions. Now, we do not need to set $k$ and the view of $g$ remains random and unpredictable on any set of queried elements (of any size). If the resulting Bloom filter is not a BP-test resilient, then this test can be used to distinguish the PRF from a truly random function. We conclude with the following theorem that we have just proved:

**Theorem 11.** *Assuming the existence of one-way functions, then for any $n \in \mathbb{N}$, universe of size $n < u \in \mathbb{N}$, and $0 < \varepsilon < 1/2$ there exists a Bloom filter that is an $(n, \varepsilon)$-strongly BP test resilient and uses $O\big(n \log \frac{1}{\varepsilon} + \lambda\big)$ bits of memory. In fact, let $\boldsymbol{B}'$ be a Bloom filter as constructed above. Then for any constant $0 < \varepsilon < 1/2$, $\boldsymbol{B}'$ is an $(n, \varepsilon)$-strongly BP test resilient and uses $O\big(n \log \frac{1}{\varepsilon} + \lambda\big)$ bits of memory.*

Note that it is not known whether replacing the hash functions with a PRF in the standard construction of Bloom filters (i.e. the one in the style of Bloom's original one [4]) results in a Bloom filter that is BP test resilient.

**Acknowledgments.** We would like to thank the anonymous reviewers for their insightful comments and suggestions.

## References

1. Ben-Eliezer, O., Jayaram, R., Woodruff, D.P., Yogev, E.: A framework for adversarially robust streaming algorithms. SIGMOD Rec. **50**(1), 6–13 (2021)
2. Bender, M.A., Farach-Colton, M., Goswami, M., Johnson, R., McCauley, S., Singh, S.: Bloom filters, adaptivity, and the dictionary problem. In: FOCS. pp. 182–193. IEEE Computer Society (2018)
3. Berman, I., Haitner, I., Komargodski, I., Naor, M.: Hardness-preserving reductions via cuckoo hashing. J. Cryptol. **32**(2), 361–392 (2019)
4. Bloom, B.H.: Space/time trade-offs in hash coding with allowable errors. Commun. ACM **13**(7), 422–426 (1970)
5. Blum, M., Micali, S.: How to generate cryptographically strong sequences of pseudo-random bits. SIAM J. Comput. **13**(4), 850–864 (1984)

6. Boyle, E., LaVigne, R., Vaikuntanathan, V.: Adversarially robust property-preserving hash functions. In: ITCS. LIPIcs, vol. 124, pp. 16:1–16:20. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2019)
7. Carter, L., Floyd, R.W., Gill, J., Markowsky, G., Wegman, M.N.: Exact and approximate membership testers. In: STOC. pp. 59–65. ACM (1978)
8. Clayton, D., Patton, C., Shrimpton, T.: Probabilistic data structures in adversarial environments. In: CCS. pp. 1317–1334. ACM (2019)
9. Cover, T.M., Thomas, J.A.: Elements of information theory (2. ed.). Wiley (2006)
10. Dietzfelbinger, M., Woelfel, P.: Almost random graphs with simple hash functions. In: STOC. pp. 629–638. ACM (2003)
11. Earman, J.: A user's guide to the surprise exam paradoxes (July 2021), http://philsci-archive.pitt.edu/19303/
12. Fan, L., Cao, P., Almeida, J.M., Broder, A.Z.: Summary cache: a scalable wide-area web cache sharing protocol. IEEE/ACM Trans. Netw. **8**(3), 281–293 (2000)
13. Goldreich, O.: The Foundations of Cryptography - Volume 1: Basic Techniques. Cambridge University Press (2001)
14. Goldreich, O., Goldwasser, S., Micali, S.: How to construct random functions. J. ACM **33**(4), 792–807 (1986)
15. Hardt, M., Woodruff, D.P.: How robust are linear sketches to adaptive inputs? In: STOC. pp. 121–130. ACM (2013)
16. Hassidim, A., Kaplan, H., Mansour, Y., Matias, Y., Stemmer, U.: Adversarially robust streaming algorithms via differential privacy. In: NeurIPS (2020)
17. Kaplan, H., Mansour, Y., Nissim, K., Stemmer, U.: Separating adaptive streaming from oblivious streaming using the bounded storage model. In: CRYPTO (3). Lecture Notes in Computer Science, vol. 12827, pp. 94–121. Springer (2021)
18. Lee, D.J., McCauley, S., Singh, S., Stein, M.: Telescoping filter: A practical adaptive filter. In: ESA. LIPIcs, vol. 204, pp. 60:1–60:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2021)
19. Luby, M., Rackoff, C.: How to construct pseudorandom permutations from pseudorandom functions. SIAM J. Comput. **17**(2), 373–386 (1988)
20. Micciancio, D., Walter, M.: On the bit security of cryptographic primitives. In: Advances in Cryptology - EUROCRYPT 2018 Proceedings, Part I. Lecture Notes in Computer Science, vol. 10820, pp. 3–28. Springer (2018)
21. Mitzenmacher, M., Pontarelli, S., Reviriego, P.: Adaptive cuckoo filters. ACM J. Exp. Algorithmics **25**, 1–20 (2020)
22. Naor, M., Reingold, O.: On the construction of pseudorandom permutations: Luby-Rackoff revisited. J. Cryptol. **12**(1), 29–66 (1999)
23. Naor, M., Yogev, E.: Bloom filters in adversarial environments. ACM Trans. Algorithms **15**(3), 35:1–35:30 (2019 (prelim version Crpyto 2015))
24. Pagh, A., Pagh, R.: Uniform hashing in constant time and optimal space. SIAM J. Comput. **38**(1), 85–96 (2008)
25. Pagh, R.: Cuckoo hashing. In: Encyclopedia of Algorithms. Springer (2008)
26. Pagh, R., Rodler, F.F.: Cuckoo hashing. J. Algorithms **51**(2), 122–144 (2004)
27. Schrift, A.W., Shamir, A.: On the universality of the next bit test. In: CRYPTO. Lecture Notes in Computer Science, vol. 537, pp. 394–408. Springer (1990)
28. Woodruff, D.P., Zhou, S.: Tight bounds for adversarially robust streams and sliding windows via difference estimators. CoRR **abs/2011.07471** (2020)
29. Yao, A.C.: Theory and applications of trapdoor functions (extended abstract). In: FOCS. pp. 80–91. IEEE Computer Society (1982)