



מכון ויצמן למדע

WEIZMANN INSTITUTE OF SCIENCE

Thesis for the degree
Master of Science

עבודת גמר (תזה) לתואר
מוסמך למדעים

Submitted to the Scientific Council of the
Weizmann Institute of Science
Rehovot, Israel

מוגשת למועצה המדעית של
מכון ויצמן למדע
רחובות, ישראל

By
Noa Oved

מאת
נועה עובד

המר-או-דלג: פילטרי בלום עמידים בפני יריב
Bet-or-Pass: Adversarially Robust Bloom Filters

Advisors:
Prof. Moni Naor

Prof. Robert Krauthgamer

מנחים:
פרופ' מוני נאור

פרופ' רוברט קראוטגמר

February 2022

אדר א' התשפ"ב

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; however, it is unclear how one should formalize the requirement that bad events happen infrequently and those false positives are appropriately distributed. Recently, some works have been investigating this topic, suggesting different robustness definitions.

In this work, we continue 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 for an adaptive adversary. The robustness definitions are formalized by the type of test the Bloom filter should pass. We then explore the relationships between these notions and highlight the notion of *Bet-or-Pass* as capturing the desired properties of such a data structure.

Acknowledgements

First, I wish to thank Prof. Moni Naor, whose insights, refinements, and expertise contributed significantly to this research. I learned a lot from you. Second, I wish to thank Prof. Robert Krauthgamer for his help and insightful comments. Finally, I wish to thank my family for their support and love.

Contents

1	Introduction	5
1.1	Our Contributions	6
1.2	Related work	8
2	Model and Problem Definition	9
2.1	Definitions	10
3	Defining Robust Bloom filters	11
3.1	Background	11
3.2	The Many Types of Robustness	12
4	Relationships Between the Various Notions of Robustness	17
4.1	Implications	18
4.2	Separations	21
4.3	Conclusions	26
5	Computational Assumptions and One-way Functions	26
5.1	Constructions of Bet-or-Pass Resilient Bloom Filters using One-way Functions	26
5.2	Robust Bloom filters Implying One-way Functions	28
	Bibliography	29

1 Introduction

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. Bloom filters have one-sided error: for any $x \in S$, the Bloom filter must answer ‘Yes’ while for any $x \in U \setminus S$, it is allowed to answer ‘Yes’ but with a small error probability at most ε , and otherwise, it answers ‘No’. That is, it admits false positives but not false negatives.

The small memory used by Bloom filters (as opposed to storing S precisely) and the fast query time makes Bloom filters extremely useful in various areas. 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 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 space¹.

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 information is attached to each element and we wish to retrieve this information. 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 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 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 ε 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

¹The lower bound on memory requirements of a Bloom filter is $n \log 1/\varepsilon$ where n is the number of elements in S and ε is the error probability.

inputs might leak some information about the internal state of the data structure 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 [FCAB00]). 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 [NY15].

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: exists a Bloom filter satisfying this notion. Finally, it is easy to check whether a Bloom filter satisfies this definition.

Following the work of Naor and Yogev [NY15] 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 makes the Bloom filter fail if x^* is a false positive. They wanted the probability of an adversary to make the Bloom filter fail to be at most ε . 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 ε 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 fails the BP test, i.e., it is 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

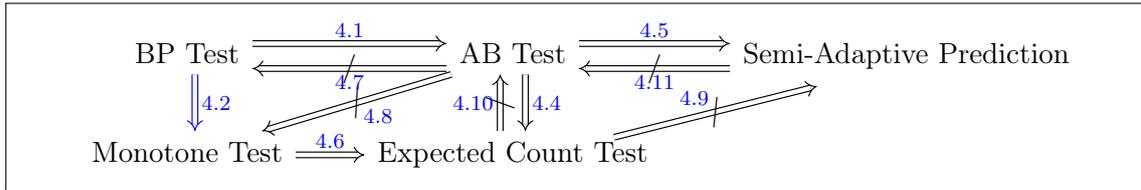


Figure 1: The relationships between the different definitions.

“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 passes 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 passes *the expected count test* if 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 chooses a set of queries Q (non-adaptively) in advance. Then she sends the elements from Q one by one to the Bloom filter and observes its results. 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 ε .

Relationships. We explore the relationships between the different definitions (see Fig. 1). We prove that a Bloom filter that passes the BP test also passes the AB test. On the other hand, we show that a Bloom filter that passes the AB test does not necessarily pass the BP test. This suggests that the BP test is a more robust notion than the AB test. We support this idea by showing that the *BP test implies monotone test resilience*² while a Bloom filter that passes the AB test is not necessarily monotone test resilient. However, we show that the AB test, in turn, implies passing the expected count test and has semi-adaptive prediction resilience. We also demonstrate that the expected count test and semi-adaptive prediction resilient are weak notions: we construct Bloom filters that satisfy those notions and fail the AB test. Finally, we show that monotone test resilience implies passing the expected count test, supporting that the expected count is indeed a special case of monotone test. We conclude that passing the BP test guarantees the desired robust properties and suggests it is the correct way to define a robust Bloom filter.

Computational Assumptions and One-way Functions. Naor and Yagev [NY15] proved existential equivalence between Bloom filters that pass the AB test (against a computationally bounded adversary) and one-way functions³. We refer to it as the equivalence result. We ask whether this equivalence still holds given a Bloom filter that passes the BP test. The simpler direction shows that a Bloom filter that passes the BP test implies the existence of one way function (we get it immediately by the implication of the BP test on the AB test). Showing

²This is reminiscent of the fact that in pseudorandomness the next-bit-test implies all efficient tests.

³One-way functions are functions that, informally speaking, are easy to compute but hard to invert.

the other direction is a little bit more challenging. We show a modification of the construction of Bloom filter from [NY15] that passes the BP test and can be based on the existence of one-way functions. This, in turn, shows the desired equivalence.

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-trivial⁴ Bloom filter fails the expected count test and can be “attacked” by an efficient adversary in a semi-adaptive prediction resilient way.

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 Yagev [NY15]. 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. Naor and Yagev also presented a tight connection between Bloom filters in their model and one-way functions, which we extend to our settings.

Following [NY15], Clayton, Patton, and Shrimpton [CPS19] 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. [BFG⁺18]. Similarly to [NY15], 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 Yagev, 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 ε . Their analysis assumes that the adversary could not find a never-queried-before element that is a false positive with probability greater than ε when using the result of previous queries. Their assumption can be achieved using the constrictions in [NY15]. Therefore, their work is orthogonal to Naor and Yagev (and ours) since their concern is dealing with repeated queries and does not deal with the problem of using adaptivity to find never-queried false positives. This case of repeated queries was also discussed by Mitzenmacher et al. [MPR20] (adaptive cuckoo filter), and by Lee et al. [LMSS21] (telescoping adaptive filter (TAF)).

The problem of defining correctness in adaptive settings was also investigated in streaming algorithms. There has been 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 [HW13] 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. [KMNS21] introduced a streaming problem that shows a gap between adversarial and

⁴Non-trivial Bloom filters are Bloom filters that require less space than the amount of space required to explicitly store the set.

oblivious streaming in the space complexity requirement. On the positive side, Ben-Eliezer et al. [BJWY21] presented generic compilers that transform a non-robust streaming algorithm into a robust one in various scenarios. Hassidim et al. [HKM⁺20] and Woodruff and Zhou [WZ20] continued their work suggesting better overhead.

2 Model and Problem Definition

We start by defining our model. For 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 the case where the set S is fixed; however, our results can be extended to other settings.

Following the work of [NY15], 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 $x \notin S$). We consider a probabilistic query algorithm 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.

Using our model we define,

Definition 2.1 (Bloom filter). Let $\mathbf{B} = (\mathbf{B}_1, \mathbf{B}_2)$ be a pair of probabilistic polynomial time algorithms such that \mathbf{B}_1 gets as input a set S and outputs a representation M , and \mathbf{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 \mathbf{B} is an (n, ε) -Bloom filter if for all sets S of size n in a suitable universe U it holds that:

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

where the probabilities are over the setup algorithm \mathbf{B}_1 and the query algorithm \mathbf{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. Definition 2.1 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, \dots, 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 $\text{AdaptiveGame}_{\mathcal{A}, t}(\lambda)$ where λ 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 S) and perform adaptive queries. \mathcal{A}_2 aims to achieve a different goal in each robustness definition. 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* λ , 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 λ . Moreover, it enables the running time of the Bloom filter to be polynomial in λ and hence the false positive probability ε can be a function of λ .

Definition 2.2. The adaptive game $\text{AdaptiveGame}_{\mathcal{A},t}(\lambda)$:

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. \mathbf{B}_1 is given input $(1^{\lambda+n \log u}, S)$ and builds a representation M .
3. The adversary \mathcal{A}_2 is given input $(1^{\lambda+n \log u}, S)$ and oracle access to $\mathbf{B}_2(M, \cdot)$ and performs at most t adaptive queries x_1, \dots, x_t to $\mathbf{B}_2(M, \cdot)$.

We assume that $x_i \notin S$ for all $i \in [t]$. This is without loss of generality since Bloom filters admit false positives, but not false negatives, and also, \mathcal{A}_2 is given as input the set S .

2.1 Definitions

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

Definition 2.3 (A test). Let \mathbf{B} be an (n, ε) -Bloom filter. For a security parameter λ , an (n, t, ε) - $\text{Test}(\lambda)_{\mathcal{A},t}$ is a game with an adversary \mathcal{A} in which \mathcal{A} needs to meet some challenge. She can choose the set S of size n and a sequence of t (mostly adaptive) queries. She gets oracle access to \mathbf{B} and tries to achieve the predefined goal.

Roughly speaking, we say that a Bloom filter \mathbf{B} passes a test if the probability (expectation) that any adversary makes it fail in the defined challenge 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 [NY15] (see subsection 1.1). However, when t is known in advance, the adversary does not have to be computationally bounded: Naor and Yegorov presented a construction of a Bloom filter that passes the AB test against computationally *unbounded* adversary using $O(n \log \frac{1}{\varepsilon} + t)$ bits of memory.

Inspired by Definition 2.5 in [NY15] we say that if \mathbf{B} passes some test for any polynomial number of queries, it *strongly passes the test*.

Definition 2.4 (Strongly passes the test). For a security parameter λ , we say that \mathbf{B} strongly passes the (n, ε) - $\text{Test}(\lambda)_{\mathcal{A},t}$, if for any polynomial $p(\cdot)$ and $t \leq p(\lambda, n)$ it holds that \mathbf{B} passes the (n, t, ε) - $\text{Test}(\lambda)_{\mathcal{A},t}$.

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. [CFG⁺78] showed that in order to construct a Bloom filter for sets of size n and error rate ε one must use (roughly) $m \geq n \log \frac{1}{\varepsilon}$ bits of memory (as opposed to $n \log u$ bits needed to answer **exact** membership queries). We can write this as $\varepsilon \geq 2^{-\frac{m}{n}}$ which leads us to the following definition:

Definition 2.5 (Minimal error (Definition 2.7 in [NY15])). Let \mathbf{B} be an (n, ε) -Bloom filter that uses m bits of memory. We say that $\varepsilon_0 = 2^{-\frac{m}{n}}$ is the minimal error of \mathbf{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 ε 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 [NY15].

Definition 2.6 (Non-trivial Bloom filter (Definition 2.8 in [NY15])). Let \mathbf{B} be an (n, ε) -Bloom filter that uses m bits of memory and let ε_0 be the minimal error of \mathbf{B} . We say that \mathbf{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 $\frac{1}{p_1(n)} < \varepsilon_0 \leq \varepsilon < 1 - \frac{1}{p_2(n)}$.⁵

Let λ be the security parameter. It holds that $n = \text{poly}(\lambda)$ ⁶, therefore we get additional upper and lower bounds: $\frac{1}{q_1(\lambda)} \leq \varepsilon_0 \leq \varepsilon \leq 1 - \frac{1}{q_2(\lambda)}$, for some polynomials $q_1(\cdot), q_2(\cdot)$.

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 ε . 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 ε 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 ε in each query. We want the Bloom filter to behave like a truly unpredictable biased coin even when an adversary sees the response of the Bloom filter on previous queries. Meaning it is 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

⁵If ε 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 [BLV19] as Lemma 4. A similar claim applies to the requirement that ε will be substantially far from 1.

⁶Because we want the adversary to run in polynomial time in the security parameter.

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 chosen beforehand queries

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

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 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 filters 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 The Many Types of Robustness

Next, we will describe five definitions of robustness for Bloom filters. In each formal definition, we outline a **test** for the Bloom filter. If it passes the test, 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.

3.2.1 The Always-Bet (AB) Test

Our starting point is the definition of [NY15]: the adversary participates in a test and 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 $\text{ABTest}_{\mathcal{A},t}(\lambda)$

1. \mathcal{A} participate in $\text{AdaptiveGame}_{\mathcal{A},t}(\lambda)$ (Def. 2.2).
2. \mathcal{A} outputs x^* .
3. The result of the test is 1 if $x^* \notin S \cup \{x_1, \dots, x_t\}$ and $\mathbf{B}_2(M, x^*) = 1$, and 0 otherwise. If $\text{ABTest}_{\mathcal{A},t}(\lambda) = 1$, we say that \mathcal{A} makes the Bloom filter fail.

Definition 3.1 (Always-Bet (AB) Test⁷). Let $\mathbf{B} = (\mathbf{B}_1, \mathbf{B}_2)$ be an (n, ε) -Bloom filter. We say that \mathbf{B} passes the (n, t, ε) -Always-Bet (AB) test if for any probabilistic polynomial-time adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ there exists a negligible function negl such that:

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

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

3.2.2 The Bet-or-Pass (BP) Test

In Definition 3.1 the adversary **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 (similar to [CT06] chapter 6): 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 profit 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 $\text{AdaptiveGame}_{\mathcal{A},t}(\lambda)$. She outputs (b, x^*) where $x^* \notin S \cup \{x_1, \dots, x_t\}$ 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 $\text{BPTest}_{\mathcal{A},t}(\lambda)$

1. \mathcal{A} participate in $\text{AdaptiveGame}_{\mathcal{A},t}(\lambda)$ (Def. 2.2).
2. \mathcal{A} outputs (b, x^*) .
3. \mathcal{A} 's profit $C_{\mathcal{A}}$ is defined as:

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

Definition 3.2 (Bet-or-Pass (BP) Test). Let $\mathbf{B} = (\mathbf{B}_1, \mathbf{B}_2)$ be an (n, ε) -Bloom filter. We say that \mathbf{B} passes the (n, t, ε) -Bet-or-Pass (BP) test if for every probabilistic polynomial-time

⁷In [NY15] it is referred to as adversarial resilient Bloom filter.

adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ participating in $\text{BPTest}_{\mathcal{A},t}(\lambda)$, there exists a negligible function negl such that:

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

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

Note that the expected profit of an adversary outputting a random guess with probability at most ε is at most 0:

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

The BP test allows an adversary to pass (although if she wants any chance to win, 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. We support this intuition by showing that the BP test implies the AB test while the other direction does not necessarily hold. In addition, we consider another family of tests: **the monotone test**, which 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 optional in the BP test. In addition, they differ in the robustness metric: the probability of outputting a false positive vs. the expected profit. Therefore, when comparing the two, it seems that we cannot claim why they differ. However, we show that the robustness metric is equivalent (in Claim 3.3 below), meaning that allowing the adversary to pass is the reason for the difference.

Let \mathcal{A} be an adversary performing $\text{ABTest}_{\mathcal{A},t}(\lambda)$ (Def. 3.1). We can think of it as an adversary performing $\text{BPTest}_{\mathcal{A},t}(\lambda)$ (Definition 3.2) with $b = 1$ always. Therefore her expected profit is:

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

Claim 3.3. *Let \mathcal{A} be an adversary performing $\text{BPTest}_{\mathcal{A},t}(\lambda)$ with $b = 1$ always. Then there exists a negligible function negl_1 such that $\Pr[x^* \text{ is FP}] \leq \varepsilon + \text{negl}_1(\lambda)$ iff there exists a negligible function negl_2 such that $\mathbb{E}[C_{\mathcal{A}}] \leq 0 + \text{negl}_2(\lambda)$.*

Proof. Let \mathcal{A} be an adversary performing $\text{ABTest}_{\mathcal{A},t}(\lambda)$ with $b = 1$ always. Assume there exists a negligible function negl_1 such that

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

Then,

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

for some negligible function negl_2 . Now, assume there exists a negligible function negl_2 such that

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

Therefore,

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

for some negligible function negl_1 , as desired. \blacksquare

3.2.3 The Monotone Test

Recall that our wishful thinking is that a “robust” Bloom filter should behave like a truly unpredictable biased coin; Both the AB and BP tests seem unrelated to our wishful thinking. Hence, returning to our wishful thinking, we consider the monotone test. The monotone test contains $\text{AdaptiveGame}_{\mathcal{A},t}(\lambda)$ as well, but now we are not interested in any output. We examine 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 ε . 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 ε .

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. Formally,

Definition 3.4 (Monotone Function). Let $t \in \mathbb{N}$. We say that a function $f: \{0,1\}^t \rightarrow \{0,1\}$ is monotone if for every pair of neighboring strings $x, x' \in \{0,1\}^t$ that are equal in all location 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 ε , though this is hardly damaging. We are interested in clusters of false positives. This is what the **monotone** property aims to model.

Definition 3.5 (Independent Biased Sequence). Let $t \in \mathbb{N}$ and $0 < b < 1$. Consider B^t , distribution on biased sequences of length t with bias b ; that is, for every i , the probability over B^t that the i -th bit equals 1 is b . We say that B^t is a distribution of independent biased sequences if all the bits are independent, i.e., for every binary string $\alpha \in \{0,1\}^t$,

$$\Pr_{b_1 \dots b_t \sim B^t} [b_1 \dots b_t = \alpha] = b^k \cdot (1-b)^{t-k},$$

where $0 \leq k \leq t$ is the number of 1's in α .

Definition 3.6 (Polynomial Indistinguishability). Two distribution S_1 and S_2 are polynomially indistinguishable if, for every probabilistic polynomial-time algorithm (distinguisher) $D: \{0,1\}^t \rightarrow \{0,1\}$,

$$\left| \Pr_{S_1}[D = 1] - \Pr_{S_2}[D = 1] \right| \leq \text{negl}(t).$$

Let $f: \{0,1\}^t \rightarrow \{0,1\}$ be a monotone function such that there exists a polynomial time algorithm D_f 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 pass *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 ε .

Definition 3.7 (Monotone Test Resilient). Let $\mathbf{B} = (\mathbf{B}_1, \mathbf{B}_2)$ be an (n, ε) -Bloom filter. We say that \mathbf{B} is (n, t, ε) -monotone test resilient if for every monotone probabilistic polynomial-time algorithm (distinguisher) $D : \{0, 1\}^t \rightarrow \{0, 1\}$ and every probabilistic polynomial-time adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ participating in $\text{AdaptiveGame}_{\mathcal{A}, t}(\lambda)$ there exists a negligible function negl such that:⁸

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

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

Note the similarity and differences with the definition of cryptographic pseudorandomness (see [Gol01]). Here, we consider only monotone tests and we look at the difference between the probabilities without an absolute value.

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,

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

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.

3.2.4 The Expected Count Test

We use $\text{AdaptiveGame}_{\mathcal{A}, t}(\lambda)$. Similar to the monotone test, we are not interested in any output. Inspired by [CPS19] 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 $\text{AdaptiveGame}_{\mathcal{A}, t}(\lambda)$ (Def. 2.2). Let $Q = \{x_1, \dots, x_t\}$ be the queries performed by \mathcal{A} . Denote the number of false positive queries by $\#\text{FP}_t := |\{x_i \mid \mathbf{B}_2(M, x_i) = 1 \text{ and } x_i \in Q \setminus S\}|$.⁹ We want to upper bound the expected number of false positives queries. Formally,

Definition 3.8 (Expected Count Test). Let $\mathbf{B} = (\mathbf{B}_1, \mathbf{B}_2)$ be an (n, ε) -Bloom filter. We say that \mathbf{B} passes the (n, t, ε) -expected count test if for any probabilistic polynomial-time adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ participating in $\text{AdaptiveGame}_{\mathcal{A}, t}(\lambda)$ there exists a negligible function negl such that:

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

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

3.2.5 The Semi-Adaptive Prediction Test

Finally, we define a semi-adaptive test: the adversary chooses the queries in advance (non-adaptively) but 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.

⁸With exactly t queries.

⁹Note that we count the number of false positives without duplicates to not over-credit the adversary.

We formalize this by defining a game, $\text{SemiAdaptiveGame}_{\mathcal{A},t}(\lambda)$. This is done in a fashion similar to $\text{AdaptiveGame}_{\mathcal{A},t}(\lambda)$, 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 t distinct queries x_1, \dots, x_t , and \mathcal{A}_2 gets as input the set S , the queries x_1, \dots, x_t and oracle access to the query algorithm (initialized with S). \mathcal{A}_2 aims to find a false positive element among the t queries without querying this element explicitly.

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

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, \dots, x_t .¹⁰
2. \mathbf{B}_1 is given input $(1^{\lambda+n \log u}, S)$ and builds a representation M .
3. The adversary \mathcal{A}_2 is given input $(1^{\lambda+n \log u}, S, (x_1, \dots, x_t))$ 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 $\text{SemiAdaptiveGame}_{\mathcal{A},t}(\lambda) = 1$, we say that \mathcal{A} makes the Bloom filter fail.

Definition 3.9 (Semi-Adaptive Prediction Resilient). Let $\mathbf{B} = (\mathbf{B}_1, \mathbf{B}_2)$ be an (n, ε) -Bloom filter. We say that \mathbf{B} is an (n, t, ε) -semi adaptive prediction resilient Bloom filter if for every probabilistic polynomial-time adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ there exists a negligible function negl such that:

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

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 [BFG⁺18] where the Bloom filter false-positive rate has to be at most ε 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.

¹⁰For 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 .

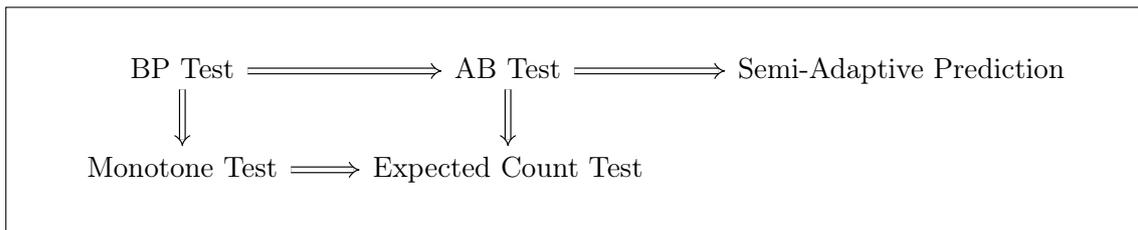


Figure 2: Tests' Implications

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 Test_1 implies Test_2 then a Bloom filter that strongly passes Test_1 also strongly passes Test_2 . We present our results considering a polynomial-time adversary; however, they also apply against unbounded adversaries if t is known in advance.

4.1.1 The BP Test

Theorem 4.1. *Let $0 < \varepsilon < 1$ and $n \in \mathbb{N}$. Let \mathbf{B} be a Bloom filter that strongly passes the (n, ε) -BP test. Then \mathbf{B} strongly passes the (n, ε) -AB test.*

Proof. Appears in the proof of Claim 3.3. ■

Theorem 4.2. *Let $0 < \varepsilon < 1$ and $n \in \mathbb{N}$. Let \mathbf{B} be a Bloom filter that strongly passes the (n, ε) -BP test. Then \mathbf{B} is strongly (n, ε) -monotone test resilient.*

Proof. Suppose, towards contradiction that \mathbf{B} is not strongly (n, ε) -monotone test resilient. There exists a monotone probabilistic polynomial-time test D and probabilistic polynomial-time adversary \mathcal{A} performing $t \leq \text{poly}(\lambda, n)$ queries such that D distinguish $D_{\mathcal{A}}$ from the biased independent sequence B_{ε} ; that is, for some polynomial p and infinitely many λ 's,

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

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

Claim 4.3. *There exists $i^* \in \{1, \dots, t\}$ such that*

$$\Pr_{S \in H_{\lambda}^{i^*}} [D(S) = 1] - \Pr_{S \in H_{\lambda}^{i^*-1}} [D(S) = 1] \geq \frac{1}{p(\lambda) \cdot t}. \quad (2)$$

Proof. The proof is immediate by Eq.(1), the pigeonhole principle and the definition of the hybrids. In particular, we use the fact that $H_{\lambda}^t = D_{\mathcal{A}}$ 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—once 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, \dots, x_{i-1} . For each $j \in [i - 1]$ let $y_j = \mathbf{B}(S, x_j)$.
3. Select r_{i+1}, \dots, r_t independently with bias ε in $\{0, 1\}$ ($\Pr[r_j = 1] = \varepsilon$).
4. If $D(y_1, \dots, y_{i-1}, 1, r_{i+1}, \dots, r_t) \neq D(y_1, \dots, y_{i-1}, 0, r_{i+1}, \dots, r_t)$, 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)

Define the set of the sequences

$$\text{BET}_i = \{y_1, \dots, y_{i-1}, r_{i+1}, \dots, r_t \mid D(y_1, \dots, y_{i-1}, 1, r_{i+1}, \dots, r_t) \neq D(y_1, \dots, y_{i-1}, 0, r_{i+1}, \dots, r_t)\}$$

and

$$\text{ONE}_i = \{y_1, \dots, y_{i-1}, r_{i+1}, \dots, r_t \mid D(y_1, \dots, y_{i-1}, 1, r_{i+1}, \dots, r_t) = D(y_1, \dots, y_{i-1}, 0, r_{i+1}, \dots, r_t) = 1\}$$

By the monotonicity of D we have that if

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

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

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

and,

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

where the probabilities are over the internal randomness of \mathbf{B} and \mathcal{A} and the biased coin flips. Combining the above with Eq.(2) we get:

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

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

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

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

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

Therefore the expected profit of \mathcal{A}_{BP} is noticeably greater than 0. ■

4.1.2 The AB Test

Theorem 4.4. *Let $0 < \varepsilon < 1$ and $n \in \mathbb{N}$. Let \mathbf{B} be a Bloom filter that strongly passes the (n, ε) -AB test. Then \mathbf{B} strongly passes the (n, ε) -expected count test.*

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

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

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, ε) -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

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

Then,

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

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

Theorem 4.5. *Let $0 < \varepsilon < 1$ and $n \in \mathbb{N}$. Let \mathbf{B} be a Bloom filter that strongly passes the (n, ε) -AB test. Then \mathbf{B} is strongly (n, ε) -semi-adaptive prediction resilient.*

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

4.1.3 The Monotone Test

Theorem 4.6. *Let $0 < \varepsilon < 1$ and $n \in \mathbb{N}$. Let \mathbf{B} be a strongly (n, ε) -monotone test resilient Bloom filter. Then \mathbf{B} strongly passes the (n, ε) -expected count test.*

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

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

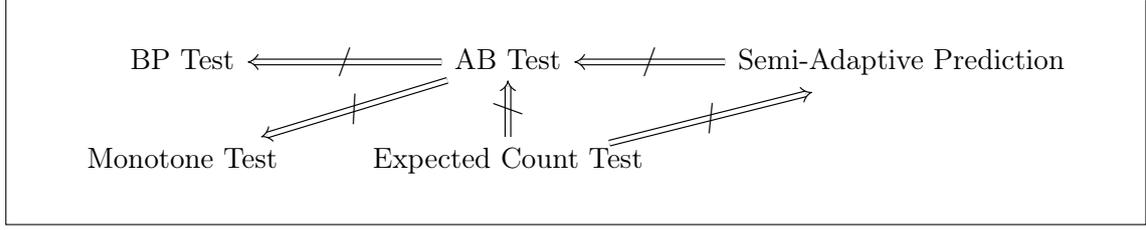


Figure 3: Separations

Therefore, there must exist $1 \leq j \leq t$ s.t. for infinitely many λ '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 $t \leq \text{poly}(\lambda, n)$. 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 strongly (n, ε) -monotone test resilient. Indeed,

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

as desired. ■

4.2 Separations

Next, we show separations between the different tests (see Fig. 3); that is, to show a separation between Test_1 and Test_2 we present a construction of a Bloom filter that strongly passes Test_1 but does not strongly pass Test_2 .

4.2.1 AB Test

Theorem 4.7. *Let $0 < \varepsilon < 1$ and $n \in \mathbb{N}$.*

1. *Assuming the existence of one-way functions, then there exists a non-trivial Bloom filter \mathbf{B} that strongly passes the (n, ε) -AB test and does not strongly pass the (n, δ) -BP test for any $0 < \delta < 1$.*
2. *If t is known in advance, then there exists a non-trivial Bloom filter \mathbf{B} that passes the (n, t, ε) -AB test and does not pass the (n, t, δ) -BP test for any $0 < \delta < 1$.*

Proof. Let $0 < \varepsilon_1 < 1$, $n \in \mathbb{N}$. First, assume that t is unknown and unbounded. To have a Bloom filter that strongly passes the (n, ε_1) -AB test, we need to use Naor and Yogev construction that uses one-way functions—using that, and we have a Bloom filter \mathbf{B} that strongly passes the (n, ε_1) -AB test. Let $0 < \varepsilon_2 < 1$. 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 ε_2 to decide whether it always answers 1 (regardless of the input) or always answers as \mathbf{B} . Let $\varepsilon = \varepsilon_2 + (1 - \varepsilon_2) \cdot \varepsilon_1$. The probability of false positive in \mathbf{B}' equals ε ; meaning \mathbf{B}' is an (n, ε) -Bloom filter. Moreover, for all $t \leq \text{poly}(\lambda, n)$, \mathbf{B}' passes the (n, t, ε) -AB test:

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

where in the inequality we used the fact that \mathbf{B} strongly passes the (n, ε_1) -AB test. We show that \mathbf{B} does not strongly pass the (n, δ) -BP test for any $0 < \delta < 1$. Let $t \leq \text{poly}(\lambda, n)$. We describe an adversary \mathcal{A} that causes \mathbf{B}' to fail in the (n, t, δ) -BP test. \mathcal{A} queries random elements in $U \setminus S$ to check if 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:

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

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 δ . In this case, the profit is non-negative.
3. Otherwise, the profit is negative, but this happens with probability δ^t (which is a very small probability for sufficiently large t)¹¹.

Summing up the above cases, we get that the expected profit of \mathcal{A} is noticeably greater than 0, meaning \mathbf{B} does not strongly pass the (n, δ) -BP test.

If t is known in advance, we use Naor and Yegorov construction of Bloom filter that passes the (n, t, ε_1) -AB test against unbounded adversaries. The above proof holds for this specific t . ■

Theorem 4.8. *Let $0 < \varepsilon < 1$ and $n \in \mathbb{N}$.*

¹¹Any (n, ε) Bloom filter is robust when the number of queries is small. Therefore, if t is not large enough, it is not interesting.

1. Assuming the existence of one-way functions, then there exists a non-trivial Bloom filter \mathbf{B} that strongly passes the (n, ε) -AB test and is not strongly (n, δ) -monotone test resilient for any $0 < \delta < 1$.
2. If t is known in advance, then there exists a non-trivial Bloom filter \mathbf{B} that passes the (n, t, ε) -AB test and is not strongly (n, t, δ) -monotone test resilient for any $0 < \delta < 1$.

Proof. Let $0 < \varepsilon_1 < 1$, $0 < \varepsilon_2 < 1$ and $0 < \delta < 1$. We use the Bloom filter \mathbf{B}' defined in the proof of Theorem 4.7. We show that \mathbf{B}' is not monotone test resilient by proving it fails the **FP's count distinguisher**, denoted by D_w (for some $w < t$). Recall,

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

We define 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 $x_i \in U \setminus S$.

Observe that with probability at least ε_2 \mathcal{A} can achieve as many false positives as she wants (w.p. ε_2 all the queries are false positives). Meaning for any $t \leq \text{poly}(\lambda, n)$ we have

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

On the other hand,

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

Therefore,

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

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

4.2.2 Expected Count Test

Theorem 4.9. *There exists a Bloom filter \mathbf{B} that strongly passes the (n, ε) -expected count test, for some $0 < \varepsilon < 1$, $n \in \mathbb{N}$, and is not strongly (n, δ) -semi-adaptive prediction resilient for any $0 < \delta < 1$.*

Proof. Let $0 < \varepsilon_1 < 1$, $n \in \mathbb{N}$ and $0 < \delta < 1$. 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 ε_1 for random queries). In order to determine which element is positive in each

set, we use a pseudorandom function¹² PRF. The PRF gets as input the block name and outputs the positive element in the block.

Set $\varepsilon = \frac{2\varepsilon_1}{1+\varepsilon_1}$. As we shall see, the resulting Bloom filter \mathbf{B} strongly passes the (n, ε) -expected count test. 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):

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

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 εt , as desired.

On the other hand, \mathbf{B} is not a strongly (n, ε) -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,

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

for sufficiently large λ and some polynomials $q(\cdot), s(\cdot)$. ■

Theorem 4.10. *There exists a Bloom filter \mathbf{B} that strongly passes the (n, ε) -expected count test, for some $0 < \varepsilon < 1$, $n \in \mathbb{N}$, and does not strongly pass the (n, ε) -AB test.*

Proof. Same example as in the proof of Theorem 4.9. ■

¹²A pseudorandom function (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.

4.2.3 Semi-adaptive Prediction Resilient

Theorem 4.11. *Let $0 < \varepsilon < 1$ and $n \in \mathbb{N}$.*

1. *Assuming the existence of one-way functions, then there exists a non-trivial Bloom filter \mathbf{B} that is strongly (n, ε) -semi-adaptive prediction resilient and does not strongly pass the (n, δ) -AB test for any $0 < \delta < 1$.*
2. *If t is known in advance and sufficiently large, then there exists a non-trivial Bloom filter \mathbf{B} that is (n, t, ε) -semi-adaptive prediction resilient and does not pass the (n, t, δ) -AB test for any $0 < \delta < 1$.*

Proof. We construct a Bloom filter \mathbf{B} that can only be broken using adaptive queries. \mathbf{B} stores one element x_{FP} to be a false positive. We want to be able to recover x_{FP} using adaptive queries. We do this in the following way. First, partition the universe into blocks of size $b = b(\varepsilon) > 4n/\varepsilon$; we define a block by its b elements x_1, \dots, x_b (the blocks are ordered). We assume the partition is public. Each block encodes a bit in $\{0, 1\}$ and we want some $\log u$ blocks to encode x_{FP} . We make sure that adaptivity only can help with finding x_{FP} .

Let $0 < \varepsilon < 1$ and $n \in \mathbb{N}$. Let $\hat{\mathbf{B}}$ be a Bloom filter that strongly passes the $(n, \varepsilon/2)$ -AB test (and by Theorem 4.5 is also strongly $(n, \varepsilon/2)$ -semi-adaptive prediction resilient), using the appropriate [NY15] construction. We build a new Bloom filter \mathbf{B} using $\hat{\mathbf{B}}$. Let $S \subset U$, $|S| = n$ and let $\alpha = \{x_1, \dots, x_b\}$ be a block. We look at $\hat{\mathbf{B}}(S, x_1), \dots, \hat{\mathbf{B}}(S, x_b)$ and count the number of positives (ones). We define that the block α encodes 1 or 0 according to the number of positives in it: if

$$\varepsilon/2 \cdot b \leq \#1\text{'s in } \alpha \leq 3\varepsilon/4 \cdot b$$

then it encodes 1 and 0 otherwise.

We look at the first $\log u$ blocks. We choose an element $x_{\text{FP}} \notin S$ that does not belong to these first blocks. We store it in \mathbf{B} to be a false positive. We want the first $\log u$ blocks to encode x_{FP} . In order to do that, we add “synthetic” false-positive elements: we change some of the *true negatives* to be *false positives*. We need to make sure that a block contains at most $3\varepsilon/4 \cdot b$ ones with high probability (to make it possible to change a block that encodes 0 to encode 1). This holds (with high probability) since $4n/\varepsilon < b \iff n < \varepsilon/4 \cdot b$. We store the modified output of the synthetic noise added in the first $\log u$ blocks.

For the query algorithm, on input x , we first check if x belongs to the first $\log u$ blocks. If it does not, then we output $\hat{\mathbf{B}}(S, x)$. Otherwise, we check if it was part of the synthetic noise we add (this must be stored) and we output the value accordingly.

Notice that the only additional memory we need is storing the output on the elements in the first $\log u$ blocks which takes at most $\varepsilon/2 \cdot b \cdot \log u$ bits¹³ and to store x_{FP} which takes $\log u$ bits. \mathbf{B} is an (n, ε) -Bloom filter (in each block we changed at most $\varepsilon/2 \cdot b$ elements.). With $t = b \cdot \log u$ an adversary can find the element x_{FP} that is known to be a false positive before querying: she simply queries the first $\log u$ blocks that encode x_{FP} and outputs x_{FP} . Meaning, \mathbf{B} does not pass the (n, t, δ) -AB test. However, \mathbf{B} is strongly (n, ε) -semi-adaptive prediction resilient since, in that case, the adversary chooses the queries non-adaptively, so it will not know the encoding of x_{FP} . ■

¹³Using pseudorandomness we can shrink it even further.

4.3 Conclusions

We showed that if a Bloom filter passes the BP test, it passes any monotone test. At the same time, this does not necessarily hold for a Bloom filter that passes 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 conclude that passing the BP test guarantees more robust properties, and therefore we suggest using it as a robustness definition.

5 Computational Assumptions and One-way Functions

5.1 Constructions of Bet-or-Pass Resilient Bloom Filters using One-way Functions

What we know so far:

$$\text{The BP Test} \begin{array}{c} \xrightarrow{\text{We showed}} \\ \xleftarrow{\text{[NY15]}} \end{array} \text{The AB Test} \xleftarrow{\text{OWF}}$$

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 Bloom filters that passes the BP test. For that, we show a construction of a Bloom filter that strongly passes the BP test 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 ([GGM86]), 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 [LR88], [NR99]).

Constructing Bloom filters That Pass The BP Test. Our starting point is the transformation presented by Naor and Yagev. As mentioned before, they showed that any Bloom filter could be efficiently transformed into a Bloom filter that strongly passes the (n, ε) -AB test 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 a Bloom filter that strongly passes the (n, ε) -BP test. 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 4.7). Applying a pseudorandom permutation, in that case, will make no difference; the adversary presented in separation 4.7 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., $|u| = \text{poly}(\lambda)$ 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^* . Now, based on the number of false positives she has seen so far, she knows with high probability if x^* is false positive or not and chooses to bet or pass accordingly.

Therefore, we cannot use the transformation of Naor and Yegorov when constructing a Bloom filter that passes the BP test.¹⁴

Apart from the above mentioned transformation, Naor and Yegorov presented a construction of a Bloom filter \mathbf{B} that passes the (n, t, ε) -AB test 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} also passes the (n, t, ε) -BP test 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 Yegorov's construction (building on Carter et al. [CFG⁺78]). They suggested to use a Cuckoo Hashing implementation of dictionary ([PR04], [Pag08]) 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 that location a function $g(x)$ is stored at either $T_1[h_1(x)]$ or $T_2[h_2(x)]$. When doing a lookup of x the values stored in $T_1[h_1(x)]$ and $T_2[h_2(x)]$ are compared to $g(x)$ and a yes is returned iff at least one of them is equal to $g(x)$. The range of g is about $\frac{2}{\varepsilon}$.

For the hash function $g: U \mapsto V$, 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 [PP08], and [DW03]). Note that the guarantee of the function still holds even when the set of queries is chosen adaptively [BHK^N19]. To reduce the memory size further, they use the family G slightly differently. Let $\ell = O(\log \frac{1}{\varepsilon})$, and set $k = O(\frac{t}{\ell})$. 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 ℓ 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 Yegorov showed that even though the adversary performs t queries, each of the ℓ 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 passes the AB test for t queries and uses $O(n \log \frac{1}{\varepsilon} + t)$ 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 (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 proportional to 2 over the range size which is ε .

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.

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

¹⁴There may be various approaches to modify the transformation; for instance, test if we are in case 1 and reselect the random bits or combat case 2 by adding as noise false positives.

To get a Bloom filter \mathbf{B}' that strongly passes the (n, ε) -BP test, 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 on any set of queried elements (of any size). If the resulting Bloom filter does not pass the BP-test, 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 5.1. *For any $n \in \mathbb{N}$, and $0 < \varepsilon < 1/2$ there exists a Bloom filter that strongly passes the (n, ε) -BP test and uses $O(n \log \frac{1}{\varepsilon} + \lambda)$ bits of memory. In fact, let \mathbf{B}' be a Bloom filter as constructed above. Then for any constant $0 < \varepsilon < 1/2$, \mathbf{B}' strongly passes the (n, ε) -BP test and uses $O(n \log \frac{1}{\varepsilon} + \lambda)$ bits of memory.*

Note that it is not known if replacing the hash functions with a PRF in the standard construction of Bloom filters results in a Bloom filter that passes the BP test.

5.2 Robust Bloom filters Implying One-way Functions

Naor and Yaguev showed that the existence of Bloom filters that strongly pass the AB test implies the existence of one-way functions. Specifically, they showed that if one-way functions do not exist, then any Bloom filter can be “attacked” (fail in the AB test) with high probability. We ask whether the weaker notion of robustness implies one-way functions. We show that even the weaker notions of the expected count test and semi-adaptive prediction resilience imply one-way functions. Our proofs rely on the proof in [NY15] (Theorem 4.1). We suggest some changes to algorithm `Attack`. We show that with these changes, any Bloom filter does not strongly pass the expected count test and is not strongly semi-adaptive prediction resilient (considering different changes for each case). The changes are marked in [blue](#).

Theorem 5.2. *Let $\mathbf{B} = (\mathbf{B}_1, \mathbf{B}_2)$ be any non-trivial Bloom filter of n elements that uses m bits of memory and let ε_0 be the minimal error of \mathbf{B} . If one-way functions do not exist, then for any constant $\varepsilon < 1$, \mathbf{B} is not (n, t, ε) -expected count test for $t = O(m/\varepsilon_0)$.*

Proof. Similar to Naor and Yaguev, we assume $\varepsilon < 2/3$ (for other values of ε the same proof still works with an appropriate adjustment of the constants).

The Algorithm Attack

1. For $i \in [t]$ sample $x_i \in U$ uniformly at random and query $y_i = \mathbf{B}_2(M, x_i)$.
2. Run A on $(x_1, \dots, x_t, y_1, \dots, y_t)$ and 1^λ to get an inverse $(S', r', x_1, \dots, x_t)$.
3. Compute $M' = \mathbf{B}_1(S', r')$.
4. Do $s = \frac{10(\frac{2}{3} - \varepsilon_0)t}{\frac{19}{20} - \frac{2}{3}}$ times:
 - (a) Do $k = \frac{200}{\varepsilon_0}$ times:
 - i. Sample $x \in U \setminus \{x_1, \dots, x_t\} \cup S$ uniformly at random.
 - ii. If $\mathbf{B}_2(M', x) = 1$ query x .
 - (b) Query an arbitrary $x \in U \setminus \{x_1, \dots, x_t\} \cup S$.

We have:

$$\mathbb{E}[\#\text{FP}_{(t+s)}] \stackrel{(*)}{\geq} \varepsilon_0 \cdot t + s \cdot \frac{19}{20} \stackrel{(**)}{>} \frac{2}{3} \cdot (s+t) > \varepsilon \cdot (s+t)$$

$$(*) : \text{From [NY15]: } \Pr[x \text{ is not FP}] \leq \frac{4}{100} + 2^{-n} \underbrace{\leq}_{n \geq 7} \frac{1}{20}$$

$$(**) : \varepsilon_0 \cdot t + s \cdot \frac{19}{20} > \frac{2}{3} \cdot (s+t) \iff \left(\varepsilon_0 - \frac{2}{3}\right) \cdot t + \left(\frac{19}{20} - \frac{2}{3}\right) \cdot s > 0 \iff s > \frac{\left(\frac{2}{3} - \varepsilon_0\right) \cdot t}{\frac{19}{20} - \frac{2}{3}}$$

■

Theorem 5.3. *Let $\mathbf{B} = (\mathbf{B}_1, \mathbf{B}_2)$ be any non-trivial Bloom filter of n elements that uses m bits of memory and let ε_0 be the minimal error of \mathbf{B} . If one-way functions do not exist, then for any constant $\varepsilon < 1$, \mathbf{B} is not (n, t, ε) -semi adaptive prediction resilient for $t = O(m/\varepsilon_0)$.*

Proof. The Algorithm Attack

1. For $i \in [t + \frac{200}{\varepsilon_0}]$ sample $x_i \in U$ uniformly at random. For $i \in [t]$, query $y_i = \mathbf{B}_2(M, x_i)$.
2. Run A on $(x_1, \dots, x_t, y_1, \dots, y_t)$ and 1^λ to get an inverse $(S', r', x_1, \dots, x_t)$.
3. Compute $M' = \mathbf{B}_1(S', r')$.
4. For $k = 1, \dots, \frac{200}{\varepsilon_0}$:
 - (a) If $\mathbf{B}_2(M', x_{t+k}) = 1$ output $t+k$ and HALT.
5. Output an arbitrary $i \in [t+1, t + \frac{200}{\varepsilon_0}]$.

The correctness of the theorem follows immediately from the proof in [NY15].

■

References

- [BFG⁺18] Michael A. Bender, Martin Farach-Colton, Mayank Goswami, Rob Johnson, Samuel McCauley, and Shikha Singh. Bloom filters, adaptivity, and the dictionary problem. In *FOCS*, pages 182–193. IEEE Computer Society, 2018.
- [BHKN19] Itay Berman, Iftach Haitner, Ilan Komargodski, and Moni Naor. Hardness-preserving reductions via cuckoo hashing. *J. Cryptol.*, 32(2):361–392, 2019. doi:10.1007/s00145-018-9293-0.
- [BJWY21] Omri Ben-Eliezer, Rajesh Jayaram, David P. Woodruff, and Eylon Yogev. A framework for adversarially robust streaming algorithms. *SIGMOD Rec.*, 50(1):6–13, 2021.
- [BLV19] Elette Boyle, Rio LaVigne, and Vinod Vaikuntanathan. Adversarially robust property-preserving hash functions. In *ITCS*, volume 124 of *LIPICs*, pages 16:1–16:20. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019.
- [CFG⁺78] Larry Carter, Robert W. Floyd, John Gill, George Markowsky, and Mark N. Wegman. Exact and approximate membership testers. In *STOC*, pages 59–65. ACM, 1978.

- [CPS19] David Clayton, Christopher Patton, and Thomas Shrimpton. Probabilistic data structures in adversarial environments. In *CCS*, pages 1317–1334. ACM, 2019.
- [CT06] Thomas M. Cover and Joy A. Thomas. *Elements of information theory (2. ed.)*. Wiley, 2006.
- [DW03] Martin Dietzfelbinger and Philipp Woelfel. Almost random graphs with simple hash functions. In *STOC*, pages 629–638. ACM, 2003.
- [FCAB00] Li Fan, Pei Cao, Jussara M. Almeida, and Andrei Z. Broder. Summary cache: a scalable wide-area web cache sharing protocol. *IEEE/ACM Trans. Netw.*, 8(3):281–293, 2000. doi:10.1109/90.851975.
- [GGM86] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions. *J. ACM*, 33(4):792–807, 1986. doi:10.1145/6490.6503.
- [Gol01] Oded Goldreich. *The Foundations of Cryptography - Volume 1: Basic Techniques*. Cambridge University Press, 2001.
- [HKM⁺20] Avinatan Hassidim, Haim Kaplan, Yishay Mansour, Yossi Matias, and Uri Stemmer. Adversarially robust streaming algorithms via differential privacy. In *NeurIPS*, 2020.
- [HW13] Moritz Hardt and David P. Woodruff. How robust are linear sketches to adaptive inputs? In *STOC*, pages 121–130. ACM, 2013.
- [KMNS21] Haim Kaplan, Yishay Mansour, Kobbi Nissim, and Uri Stemmer. Separating adaptive streaming from oblivious streaming using the bounded storage model. In *CRYPTO (3)*, volume 12827 of *Lecture Notes in Computer Science*, pages 94–121. Springer, 2021.
- [LMSS21] David J. Lee, Samuel McCauley, Shikha Singh, and Max Stein. Telescoping filter: A practical adaptive filter. In *ESA*, volume 204 of *LIPICs*, pages 60:1–60:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021.
- [LR88] Michael Luby and Charles Rackoff. How to construct pseudorandom permutations from pseudorandom functions. *SIAM J. Comput.*, 17(2):373–386, 1988. doi:10.1137/0217022.
- [MPR20] Michael Mitzenmacher, Salvatore Pontarelli, and Pedro Reviriego. Adaptive cuckoo filters. *ACM J. Exp. Algorithmics*, 25:1–20, 2020. doi:10.1145/3339504.
- [NR99] Moni Naor and Omer Reingold. On the construction of pseudorandom permutations: Luby-rackoff revisited. *J. Cryptol.*, 12(1):29–66, 1999. doi:10.1007/PL00003817.
- [NY15] Moni Naor and Eylon Yogev. Bloom filters in adversarial environments. *ACM Trans. Algorithms*, 15(3):35:1–35:30, 2019 (prelim. version Crypto 2015). doi:10.1145/3306193.
- [Pag08] Rasmus Pagh. Cuckoo hashing. In *Encyclopedia of Algorithms*. Springer, 2008.
- [PP08] Anna Pagh and Rasmus Pagh. Uniform hashing in constant time and optimal space. *SIAM J. Comput.*, 38(1):85–96, 2008. doi:10.1137/060658400.
- [PR04] Rasmus Pagh and Flemming Friche Rodler. Cuckoo hashing. *J. Algorithms*, 51(2):122–144, 2004. doi:10.1016/j.jalgor.2003.12.002.
- [WZ20] David P. Woodruff and Samson Zhou. Tight bounds for adversarially robust streams and sliding windows via difference estimators. *CoRR*, abs/2011.07471, 2020.