Bernoulli Honeywords

Ke Coby Wang
Duke University
coby.wang@ieee.org

Michael K. Reiter
Duke University
michael.reiter@duke.edu

***Abstract*—Decoy passwords, or "honeywords," planted in a credential database can alert a site to its breach if ever submitted in a login attempt. To be effective, some honeywords must appear at least as likely to be user-chosen passwords as the real ones, and honeywords must be very difficult to guess without having breached the database, to prevent false breach alarms. These goals have proved elusive, however, for heuristic honeyword generation algorithms. In this paper we explore an alternative strategy in which the defender treats honeyword selection as a Bernoulli process in which each possible password (except the user-chosen one) is selected as a honeyword independently with a fixed probability. We show how Bernoulli honeywords can be integrated into two existing system designs for leveraging honeywords: one based on a honeychecker that stores the secret index of the user-chosen password in the list of account passwords, and another that does not leverage secret state at all. We show that Bernoulli honeywords enable analytic derivation of false breach-detection probabilities irrespective of what information the attacker gathers about the sites' users; that their true and false breach-detection probabilities demonstrate compelling efficacy; and that they can even enable performance improvements in modern honeyword system designs.**

I. Introduction

In the Colonial Pipeline ransomware attack in May 2021, unauthorized access to the company network was gained via an employee's VPN account with a (complicated) password that was found in a password database breached from a different site [24], [12]. After gaining access, the attackers went on to disable part of the company's network and demanded a $5 million ransom to recover it, leading to fuel shortages across the U.S. and emergency declarations in a number of states.

This example was not an isolated incident. Breached credential databases are the source of most passwords leveraged in credential stuffing campaigns [35], which are themselves the cause of the vast majority of account takeovers [33], owing to the tendency of users to reuse passwords across sites [13], [30], [37]. Password managers (PMs) can mitigate password reuse, but they are not a panacea: A recent survey [27] at a U.S. university found that though respondents using *third-party* PMs were less likely to reuse passwords, 47% still did so. Moreover, third-party PMs were less prevalent than password-management strategies more prone to password reuse—e.g., 77% and 84% of OS- or browser-built-in PM users reported reusing passwords—yielding an overall password-reuse rate of 77%. Attackers often use leaked credentials to harvest vast numbers of accounts from sites, e.g., [31]. Discovering a credential database breach takes an average of between seven [23] and fifteen months [33], during which time the attacker can access accounts with little accountability.

To discover credential database breaches more quickly, Juels and Rivest [25] proposed that (hashes of) decoy passwords, or *honeywords*, be included alongside the user-chosen password for each account in the credential database. In this way, the attacker attempting to access an account at a site where it breached the credential database risks alerting the site to its breach, since only the attacker (and not the user) has any chance of knowing the honeywords. To be effective, however, honeyword generation faces at least two requirements. The first is *flatness*, namely that an attacker cannot reliably guess which password in the set of passwords stored for the account is the user-chosen one. The second, and arguably more critical, requirement (see Sec. II) is a quantifiable and low false-alarm rate; i.e., it must be quantifiably difficult for an attacker who has *not* breached a site's database to guess honeywords for an account at that site. The discovery of a credential database breach is an urgent, disruptive, and costly event, generally requiring that all passwords be reset and that a breach investigation commence. IBM put the average cost of a breach detection and escalation at $1.24 million [23, p. 16]. Without quantifying the risk of a false alarm and minimizing it, breach detections will be disregarded by operators.

Meeting these requirements has proved difficult. Prevailing honeyword generation techniques, which are based on heuristic methods for explicitly generating these honeywords, seem unlikely to succeed. For example, to render honeywords seemingly as likely as the user-chosen passwords to an attacker who gathers information about users (perhaps from the same database it breached to obtain the passwords), recent advice is that honeywords should include personal information [41]. However, state-of-the-art heuristics to do so come with significant risk of false breach alarms (see Sec. II). Moreover, against an attacker who knows *more* personal information about users represented in the breached database than the defender—e.g., if it identifies the same users in another breached dataset—there seems to be little hope for achieving flatness on a per-account basis. Even if the defender knows the same user information as the attacker, it may be reluctant to increase the exposure of its users' information by leveraging it to create honeywords.

Here we explore an alternative strategy that implements honeyword selection as a Bernoulli process in which *each password in the password space* (aside from the user-chosen one) is selected independently with a fixed probability to be a honeyword for this account. Intuitively, provided that these

Network and Distributed System Security (NDSS) Symposium 2024
26 February - 1 March 2024, San Diego, CA, USA
ISBN 1-891562-93-2
https://dx.doi.org/10.14722/ndss.2024.23295
www.ndss-symposium.org

honeywords sufficiently often intersect the passwords that the attacker finds at least as likely as the user-chosen password, the breach will be detected with significant probability, particularly as the number of accounts the attacker accesses grows. Of course, explicitly selecting and storing so many honeywords would be intractable, and so we instead select them implicitly, by randomly configuring a data structure that stores them.

We adapt two existing systems to use this idea: the design of Juels & Rivest [25] and a design called Amnesia [42]. A central feature of Bernoulli honeywords is that, unlike honeyword generation heuristics, they enable analytic estimation of the false and true breach-detection rates, which we provide. Critically, *the false breach-detection rate is independent of the attacker's knowledge about users' passwords*, which we argue is essential for breach detection. Moreover, Bernoulli honeywords lead to cost *improvements* in a feature of Amnesia, namely the ability for one site to monitor for the entry of its honeywords at *other* sites, which is important since stuffing breached credentials at another site enables the attacker to identify the user-chosen password if it was reused there.

To summarize, our contributions are as follows.

- We explore the use of *Bernoulli honeywords* that are not constructed but rather are sampled independently from all possible passwords. Bernoulli honeywords do not depend on honeyword-generation heuristics that, we argue, will continue to struggle against attackers who know as much or more information about a site's users than the site does.
- We describe a realization of Bernoulli honeywords in the original honeyword system design of Juels & Rivest [25]. We analyze the false- and true-detection probabilities for this construction, showing that it can be highly effective as a breach-detection mechanism. In particular, the false-detection probability is independent of the attacker's knowledge about the site's users (including even their passwords), which we argue is essential.
- We describe a second realization of Bernoulli honeywords in Amnesia [42], a system design that can detect breaches without requiring that any secret state survives the breach. We analytically estimate the false- and true-detection probabilities of this design, as well; again, the former is independent of the attacker's knowledge about the site's users. We further show that our design accommodates a site monitoring for entry of its honeywords at remote sites, at an expense that is lower than in Amnesia in several important measures.

II. Background and Related Work

Honeyword system designs: We are aware of only a handful of system designs for detecting password database breaches using honeywords. We separate these proposals into two camps: *asymmetric* and *symmetric*. Asymmetric designs leverage an information asymmetry between the defender and the attacker, in the form of a secret datum that the site stores but assumes will not be captured by the attacker when he breaches the site. As we will discuss in Sec. IV, in the original proposal of Juels & Rivest [25] this information is the index of the user-chosen password within a list of passwords per account, stored in an unbreachable *honeychecker*. In Lethe [15], this secret is the seed to a pseudorandom number generator that is used to service logins within an interval of time and then later reused by an unbreachable *checking server* to detect entry of a honeyword. A third example is that of Almeshekah et al. [2], which leverages a machine-dependent function for password hashing; if the attacker who breaches the site is unaware of this design, then its attempts to crack the database offline will yield decoy passwords that the site can detect using its unbreachable machine-dependent function.

We know of only one symmetric design to date, which is Amnesia [42]. Amnesia permits the attacker to learn the entire state of a breached site. In exchange for allowing this, Amnesia enables the site to detect its breach only as legitimate users log into accounts that the attacker previously logged into. Moreover, detection is only probabilistic, though as the number of accounts the attacker logs into grows, the probability of breach detection also grows. We defer a detailed introduction of Amnesia to Sec. V.

Bernoulli honeywords can be used in both asymmetric and symmetric designs, as we will demonstrate by realizing them within both the original honeychecker design [25], which is asymmetric, and Amnesia [42], which is symmetric. In both cases, the integration reveals the need for careful additional analysis, which we provide.

Honeyword generation heuristics: A mostly distinct line of research (e.g., [17], [8], [39], [1], [41]) has developed on generating honeywords to be flat, so that the attacker cannot easily select the user-chosen password from the passwords associated with an account in the breached database. One of the most difficult aspects of ensuring flatness is that users tend to incorporate personal information in their passwords (birth year, favorite team mascot, etc.). An attacker who can mine such information about users that the site does not take into account in generating honeywords will generally be able to distinguish the user-chosen password from the decoys by selecting the one that includes personal information [39]. Recent progress has therefore advocated that personal information be incorporated into honeywords [41].

We contend that trying to match the attacker's knowledge about users in order to generate flat honeywords for them might be difficult, at best. Even a site that knows a considerable amount of personal information about its users might not wish to risk further exposure of that information by importing it into the password (re)setting pipeline or results. As such, here we explore an approach different from creating a small number of explicit honeywords via tuned heuristics. Instead, our idea here (see Sec. III) is to include a fraction of all passwords as honeywords, in the hopes of there being some that the attacker finds at least as likely to be user-chosen as the actual ones.

False alarms in breach detection: The importance of a quantifiably low false-alarm rate, particularly for breach detection, was detailed in stark terms by the Tripwire study [14]. In this study, researchers worked with an email provider to monitor for logins to fake email accounts, each used to register a *decoy account* with the same password at another site. Any login to an email account suggested that the site hosting its decoy account had been breached—assuming the email provider itself had not been breached—since the only places where that password (or a hash thereof) existed were the email provider and the site hosting that decoy account. Despite DeBlasio, et al. disclosing 18 apparent site breaches (and the Tripwire methodology) to the relevant site administrators, only one-third responded at all, only one indicated that it would force a password reset, and none notified their users [14, Sec. 6.3]. DeBlasio et al. concluded, "a major open question ... is how much (probative, but not particularly illustrative) evidence produced by an external monitoring system like Tripwire is needed to convince operators to act, such as notifying their users and forcing a password reset" [14, Sec. 8]. This is compelling evidence that a quantifiable, tunable false-alarm rate is the core requirement for breach detection.

Unfortunately, honeyword generation heuristics come with significant risk of false breach alarms. For example, the best proposal of Wang et al. [41], even after blocklisting 10^5 common passwords from being selected as either honeywords or user-chosen passwords, still enables an attacker who has not breached a site to guess one of only 20 honeywords for an account with $> 6\%$ chance within only 100 online guesses [41, Fig. 4]. Wang et al. thus speculated that a threshold of three honeywords entered to raise an alarm might be more appropriate than only one, as a way to mitigate false-alarm risk. However, modern estimates of online guessing attacks (see Sec. IV-F) suggest that resilience to 100 online guesses is $10^4\times$ *too small*, raising doubts as to whether there is any suitable threshold for honeywords entered that would permit satisfactory quantification of the resulting false and true breach-detection probabilities for such heuristic approaches.

Another often overlooked subtlety is that a low false-alarm rate should be guaranteed even if an attacker learns user-chosen passwords for some accounts at the target site by other means (e.g., phishing), or even against legitimate users of the site themselves. Some compromised user-chosen passwords and a wholesale breach of the credential database should not be confused, as the reactions warranted by each are qualitatively different. Juels and Rivest [25, Sec. 7.5] propose to reduce the likelihood of false breach alarms by selecting the honeywords for an account randomly from a pool of honeywords for that user-chosen password. However, we know of no work that has demonstrated explicit generation of a honeyword pool sufficiently large to achieve a suitably low false-detection probability in the face of realistic threats as we do here, particularly against an attacker knowing user-chosen passwords. Bernoulli honeywords resolve this difficulty, ensuring a quantifiable and tunable false breach-detection probability even in this case.

III. Bernoulli Honeywords

We begin by abstractly describing Bernoulli honeywords and their properties, in terms of one user at one site. We defer constructions of systems using them to later sections.

Let $\mathsf{pwd}[r]$, $r \geq 1$, denote the list of all allowable passwords ranked in order of non-increasing likelihood for the user to have chosen at this site *from the perspective of the attacker*. Formally, if $\mathbb{U}$ is a random variable with value the rank of the password chosen by the user at this site, then for $\hat{r} < \check{r}$, $\mathbb{P}(\mathbb{U} = \hat{r}) \geq \mathbb{P}(\mathbb{U} = \check{r})$. If the attacker knows little about the user, then the $\mathsf{pwd}[\cdot]$ list might be simply a list of passwords ranked in order of popularity. However, if the attacker knows personal information about the user, then this order might reflect that personal information. We stress that the defender (the site) will generally not know the distribution of $\mathbb{U}$.

At its core, our key idea is simple: Suppose that during password (re)set, each possible password other than the one the user chose is selected independently with probability p_{h} as a honeyword for the user at this site. Per account, let $\mathbb{S}[r]$ be an indicator random variable such that $\mathbb{S}[r] = 1$ if and only if either $\mathbb{U} = r$ or $\mathsf{pwd}[r]$ is chosen as a honeyword (with probability p_{h}), and $\mathbb{S}[r] = 0$ otherwise. We explore the implications of this idea to two attackers.

A. *Raising alarm attacker (raat)*

We first consider an attacker that does *not* breach the site but that wishes to enter a honeyword to induce a (false) alarm at it. In this threat model, we permit the *raat* to know the user-chosen password and so its rank. While this permits the *raat* to access the account as the intended user could—indeed, the *raat* might *be* the intended user—that is not our concern here. Rather, our concern is the *raat*'s ability to input a honeyword despite no breach having occurred.

Let $\textsc{hwin}^{raat}(p_{\mathsf{h}}, \ell)$ denote the probability with which at least one honeyword is input by a *raat* when it attempts ℓ distinct logins on one account. Since any password other than the user-chosen one that the *raat* enters in a login attempt is a honeyword with probability p_{h}, we immediately have

$$\textsc{hwin}^{raat}(p_{\mathsf{h}}, \ell) = 1 - (1 - p_{\mathsf{h}})^{\ell}$$

Note, $\textsc{hwin}^{raat}(p_{\mathsf{h}}, \ell)$ is independent of the distribution of $\mathbb{U}$.

B. *Breaching attacker (brat)*

The next attacker we consider is one who breaches the credential database for the site, thereby obtaining the per-account values $s[r]$ taken on by $\mathbb{S}[r]$ for all $r \geq 1$, and then attempts to access accounts at that site. For each account,

$$\mathbb{P}\left(\mathbb{U} = \hat{r} \;\middle|\; \textstyle\bigwedge_r \mathbb{S}[r] = s[r]\right) = \frac{\mathbb{P}\left(\mathbb{U} = \hat{r} \wedge \left(\bigwedge_r \mathbb{S}[r] = s[r]\right)\right)}{\mathbb{P}\left(\bigwedge_r \mathbb{S}[r] = s[r]\right)}$$

with the numerator being

$$\begin{aligned}&\mathbb{P}\left(\mathbb{U} = \hat{r} \wedge \left(\textstyle\bigwedge_r \mathbb{S}[r] = s[r]\right)\right)\\ &\quad = \mathbb{P}(\mathbb{U} = \hat{r}) \times \mathbb{P}\left(\textstyle\bigwedge_{r \neq \hat{r}} \mathbb{S}[r] = s[r] \;\middle|\; \mathbb{U} = \hat{r}\right)\end{aligned}$$

Now consider distinct ranks $\hat{r}$ and $\check{r}$ for which $s[\hat{r}] = s[\check{r}] = 1$. Since $\mathbb{P}\left(\bigwedge_{r \neq \hat{r}} \mathbb{S}[r] = s[r] \;\middle|\; \mathbb{U} = \hat{r}\right) = \mathbb{P}\left(\bigwedge_{r \neq \check{r}} \mathbb{S}[r] = s[r] \;\middle|\; \mathbb{U} = \check{r}\right)$, we see

$$\frac{\mathbb{P}\left(\mathbb{U} = \hat{r} \;\middle|\; \bigwedge_r \mathbb{S}[r] = s[r]\right)}{\mathbb{P}\left(\mathbb{U} = \check{r} \;\middle|\; \bigwedge_r \mathbb{S}[r] = s[r]\right)} = \frac{\mathbb{P}(\mathbb{U} = \hat{r})}{\mathbb{P}(\mathbb{U} = \check{r})}$$

In other words, observing $\{s[r]\}_{r \geq 1}$ helps the *brat* only in limiting his attention to those ranks r for which $s[r] = 1$. Among those for which $s[r] = 1$, their relative likelihoods are unchanged by $\{s[r]\}_{r \geq 1}$ from the *brat*'s perspective.

Thus, the best the *brat* can do is to try to access the account using the password with lowest rank r (i.e., the most likely password) for which $s[r] = 1$. More specifically, let

$$\mathbb{R}_\theta \overset{\text{def}}{=} \min\left\{r \geq 1 \;\middle|\; \textstyle\sum_{r'=1}^{r} \mathbb{S}[r'] = \theta\right\}$$

be a random variable denoting the minimum rank r for which there are θ passwords of at most rank r that remain possible as the user-chosen password from the *brat*'s perspective. Then, *the best move for the brat attempting to access this account without entering a honeyword is to login using* $\mathsf{pwd}[r_1]$ *where* r_1 *is the value taken on by* $\mathbb{R}_1$. When the *brat* does so, the probability of it entering the user-chosen password is

$$\mathbb{P}\left(\mathbb{U}=r_1 \;\middle|\; \bigwedge_{j=1}^{\theta}\mathbb{R}_j=r_j\right) = \frac{\mathbb{P}(\mathbb{U}=r_1)}{\sum_{j'=1}^{\theta}\mathbb{P}\left(\mathbb{U}=r_{j'}\right)+\mathbb{P}(\mathbb{U}>r_\theta)\,p_{\mathsf{h}}}$$

which is computed as the ratio of

$$\mathbb{P}\left(\mathbb{U}=r_1 \wedge \bigwedge_{j=1}^{\theta}\mathbb{R}_j=r_j\right) = \mathbb{P}(\mathbb{U}=r_1)\,p_{\mathsf{h}}^{\theta-1}(1-p_{\mathsf{h}})^{r_\theta-\theta}$$

and

$$\begin{aligned}\mathbb{P}\left(\bigwedge_{j=1}^{\theta}\mathbb{R}_j=r_j\right) &= \sum_{j'=1}^{\theta}\mathbb{P}\left(\mathbb{U}=r_{j'}\right)\mathbb{P}\left(\bigwedge_{j=1}^{\theta}\mathbb{R}_j=r_j \;\middle|\; \mathbb{U}=r_{j'}\right)\\ &\quad+\ \mathbb{P}(\mathbb{U}>r_\theta)\,\mathbb{P}\left(\bigwedge_{j=1}^{\theta}\mathbb{R}_j=r_j \;\middle|\; \mathbb{U}>r_\theta\right)\\ &= \sum_{j'=1}^{\theta}\mathbb{P}\left(\mathbb{U}=r_{j'}\right)p_{\mathsf{h}}^{\theta-1}(1-p_{\mathsf{h}})^{r_\theta-\theta}\\ &\quad+\ \mathbb{P}(\mathbb{U}>r_\theta)\,p_{\mathsf{h}}^{\theta}(1-p_{\mathsf{h}})^{r_\theta-\theta}\end{aligned} \tag{1}$$

Then, for $\mathcal{R}=\{r_j\}_{j=1}^{\theta}$, the probability of entering a honeyword is

$$\begin{aligned}\textsc{hwin}^{brat}(p_{\mathsf{h}},\mathcal{R}) &\overset{\text{def}}{=} \mathbb{P}\left(\mathbb{U}\neq r_1 \;\middle|\; \bigwedge_{j=1}^{\theta}\mathbb{R}_j=r_j\right)\\ &= \frac{\sum_{j'=2}^{\theta}\mathbb{P}\left(\mathbb{U}=r_{j'}\right)+\mathbb{P}(\mathbb{U}>r_\theta)\,p_{\mathsf{h}}}{\sum_{j'=1}^{\theta}\mathbb{P}\left(\mathbb{U}=r_{j'}\right)+\mathbb{P}(\mathbb{U}>r_\theta)\,p_{\mathsf{h}}}\end{aligned} \tag{2}$$

Below, we refer to an account for which $\bigwedge_{j=1}^{\theta}\mathbb{R}_j=r_j$ as an $\mathcal{R}$-account for $\mathcal{R}=\{r_j\}_{j=1}^{\theta}$.

While in general the defender will not know the distribution of $\mathbb{U}$, in the next section we will incorporate known results from previous research to estimate that distribution when evaluating the true-detection probabilities for specific system realizations based on the insights in this section.

IV. Integration with a Honeychecker

In this section we present a practical realization of the insights presented in Sec. III, which we obtain by modifying the original design of Juels & Rivest [25]. One challenge in such adaptations is finding a way to represent the honeyword status (i.e., honeyword or not) for every possible password in a compact way. As we will show, this representation can come with other consequences to the properties offered by the designs into which they are integrated.

A. Bloom filters

An ingredient of our realization below is a Bloom filter [5], which is a data structure for compactly storing a set of elements. A Bloom filter supports two operations, namely element insertion and membership testing. In brief, a Bloom filter is defined by a set $\mathcal{F}=\{f_i\}_{i=1}^{k}$ of k uniform hash functions where each $f_i:\{0,1\}^{\nu}\to\{1,\ldots,b\}$, and a set $\mathcal{B}\subseteq\{1,\ldots,b\}$, initially empty. To insert an element e into a Bloom filter $\langle\mathcal{F},\mathcal{B}\rangle$, the set $\mathcal{B}$ is updated as $\mathcal{B}\leftarrow\mathcal{B}\cup\mathcal{F}(e)$ where $\mathcal{F}(e)=\{f_i(e)\}_{i=1}^{k}$. A membership test for e, denoted $e \overset{?}{\in_{\mathrm{B}}} \langle\mathcal{F},\mathcal{B}\rangle$, returns true if and only if $\mathcal{F}(e)\subseteq\mathcal{B}$. For simplicity, we write $e\in_{\mathrm{B}}\langle\mathcal{F},\mathcal{B}\rangle$ when this test returns *true*, and $e\notin_{\mathrm{B}}\langle\mathcal{F},\mathcal{B}\rangle$ when it returns *false*. As such,

$$\mathbb{P}\left(e\in_{\mathrm{B}}\langle\mathcal{F},\mathcal{B}\rangle \;\middle|\; e\xleftarrow{\$}\{0,1\}^{\nu}\right)=\left(\frac{|\mathcal{B}|}{b}\right)^{k} \tag{3}$$

where $|\mathcal{B}|$ is the cardinality of $\mathcal{B}$ and $\xleftarrow{\$}$ denotes uniform sampling.

Note that $|\mathcal{F}(e)|<k$ if $f_i(e)=f_{i'}(e)$ for some $i\neq i'$. Below, we will leverage the facts that for $e,e'\xleftarrow{\$}\{0,1\}^{\nu}$,

$$\mathbb{E}\left(|\mathcal{F}(e)|\right)=b-b\left(1-\frac{1}{b}\right)^{k} \tag{4}$$

$$\mathbb{E}\left(\left|\mathcal{F}(e)\cup\mathcal{F}(e')\right|\right)=b-b\left(1-\frac{1}{b}\right)^{2k} \tag{5}$$

and, more importantly, the distributions of $|\mathcal{F}(e)|$ and $|\mathcal{F}(e)\cup\mathcal{F}(e')|$ are tightly concentrated around these expected values (e.g., [29, Sec. 12.5.3]). The distribution of $|\mathcal{F}(e)\setminus\mathcal{F}(e')|$, then, is tightly concentrated around

$$\mathbb{E}\left(\left|\mathcal{F}(e)\setminus\mathcal{F}(e')\right|\right)=b\left[\left(1-\frac{1}{b}\right)^{k}-\left(1-\frac{1}{b}\right)^{2k}\right] \tag{6}$$

by subtracting (4) from (5).

B. Background on honeycheckers

Juels and Rivest [25] introduced honeywords in the context of a design that detected the entry of a honeyword using a trusted component called a *honeychecker*. For each account, the site holds in its credential database a list of (hashes of) passwords, one user-chosen and the others honeywords. The index of the user-chosen password in the list is stored in the honeychecker. In a login attempt to the account using password π, the attempt fails if π (i.e., its hash) is not in the list. If π is in the list, its index in the list is sent to the honeychecker. If this index matches the index stored for this account in the honeychecker, then the login succeeds; otherwise, a breach alarm is raised.

Because the index of the user-chosen password is what enables detection of a login by a *brat*, it is necessary that the *brat* not learn the contents of the honeychecker despite breaching the site. In other words, the honeychecker must not be breachable, even if the site is.

C. Adapting a system using a honeychecker

To adapt a system leveraging a honeychecker to leverage Bernoulli honeywords, the site will store a Bloom filter per account to hold elements in $\{0,1\}^{\nu}$. The elements stored in the Bloom filter will be outputs of a password hashing function $H:\{0,1\}^{*}\to\{0,1\}^{\nu}$ modeled as a random oracle [4]. This hash function can be salted, as is standard in good password management [20]. Specifically, for an account with user-chosen password $\hat{\pi}$, the Bloom filter $\langle\mathcal{F},\mathcal{B}\rangle$ is selected first by choosing the uniform hash functions $\mathcal{F}$ randomly, and then by choosing $\mathcal{B}$ randomly subject to (i) $H(\hat{\pi})\in_{\mathrm{B}}\langle\mathcal{F},\mathcal{B}\rangle$, and (ii) $|\mathcal{B}|=(p_{\mathsf{h}})^{1/k}b$, so that (3) equals p_{h}. This Bloom filter is stored for the account at the (potentially breachable) server.

The (unbreachable) honeychecker stores the indices $\mathcal{F}(H(\hat{\pi}))$ for the account. A login attempt with password π is evaluated as follows:

$$\mathsf{login}(\pi)=\begin{cases}\textit{failure} & \text{if } H(\pi)\notin_{\mathrm{B}}\langle\mathcal{F},\mathcal{B}\rangle\\ \textit{success} & \text{if } \mathcal{F}(H(\pi))=\mathcal{F}(H(\hat{\pi}))\\ \textit{alarm} & \text{otherwise}\end{cases}$$

Note that only the honeychecker can decide between *success* or *alarm*, since only the honeychecker holds $\mathcal{F}(H(\hat{\pi}))$. That is, the server first checks for *failure* and, if its condition does not hold, the server invokes the honeychecker with $\mathcal{F}(H(\pi))$ to decide between *success* or *alarm*.

D. Security against a raat

Since a *raat* does not breach the site, the *raat* learns nothing about the Bloom filter $\langle \mathcal{F}, \mathcal{B} \rangle$ for an account except by submitting passwords in login attempts to the account. Note that the Bloom filter is configured to hold each password with probability p_{h} (except for the user-chosen one, which it holds with probability 1.0). If the *raat* has ℓ login attempts per account to enter a honeyword, and performs these attempts on n accounts, then the false-detection probability is

$$\textsc{fdp}(\ell, n) \le 1 - \left(1 - \textsc{hwin}^{raat}\left(p_{\mathsf{h}}, \ell\right)\right)^n \tag{7}$$

E. Security against a brat

Since a *brat* breaches the site, it knows the Bloom filter $\langle \mathcal{F}, \mathcal{B} \rangle$ for each account. As proved in Sec. III-B, the best the *brat* can do to login to an account is to attempt the most likely password π for which $H(\pi) \in_{\textsc{b}} \langle \mathcal{F}, \mathcal{B} \rangle$; i.e., the password $\pi = \mathsf{pwd}[r]$ for the lowest r for which $H(\mathsf{pwd}[r]) \in_{\textsc{b}} \langle \mathcal{F}, \mathcal{B} \rangle$. If π is a honeyword (which happens with probability $\textsc{hwin}^{brat}\left(p_{\mathsf{h}}, r\right)$), then the probability of the breach going undetected is the probability that $\mathcal{F}(H(\pi)) = \mathcal{F}(H(\hat{\pi}))$ for the user-chosen password $\hat{\pi}$. Note that

$$\begin{aligned} &\mathbb{P}\left(\mathcal{F}(H(\pi)) = \mathcal{F}(H(\hat{\pi})) \,\middle|\, H(\pi) \neq H(\hat{\pi})\right) \\ &\le \mathbb{P}\left(\mathcal{F}(H(\pi)) \subseteq \mathcal{F}(H(\hat{\pi})) \,\middle|\, H(\pi) \neq H(\hat{\pi})\right) \approx \left(\frac{\mathbb{E}\left(|\mathcal{F}(H(\hat{\pi}))|\right)}{|\mathcal{B}|}\right)^k \end{aligned}$$

Disregarding the possibility that $H(\pi) = H(\hat{\pi})$ even though $\pi \neq \hat{\pi}$ (which happens with probability $2^{-\nu}$), we thus estimate the true-detection probability for a *brat* attempting to login to an $\mathcal{R}$-account to be:

$$\begin{aligned} \textsc{tdp}_{\mathcal{R}} \approx\ & \textsc{hwin}^{brat}\left(p_{\mathsf{h}}, \mathcal{R}\right) \times \\ & \mathbb{P}\left(\mathcal{F}(H(\pi)) \neq \mathcal{F}(H(\hat{\pi})) \,\middle|\, H(\pi) \neq H(\hat{\pi})\right) \\ \gtrapprox\ & \textsc{hwin}^{brat}\left(p_{\mathsf{h}}, \mathcal{R}\right) \times \left(1 - \left(\frac{\mathbb{E}\left(|\mathcal{F}(H(\hat{\pi}))|\right)}{|\mathcal{B}|}\right)^k\right) \end{aligned} \tag{8}$$

with the expected value instantiated as in (4). Let $\mathcal{R}_a$ be the rank set such that account a is an $\mathcal{R}_a$-account. The true-detection probability for a *brat* who attacks accounts $\mathcal{A}' \subseteq \mathcal{A}$, where $\mathcal{A}$ is the set of accounts at the breached site, is

$$\textsc{tdp}(\mathcal{A}') = 1 - \prod_{a \in \mathcal{A}'} \left(1 - \textsc{tdp}_{\mathcal{R}_a}\right) \tag{9}$$

and the *minimum* true-detection probability for *brat* who attacks n accounts is

$$\textsc{tdp}(n) = \min_{\mathcal{A}' \subseteq \mathcal{A} : |\mathcal{A}'| = n} \textsc{tdp}(\mathcal{A}') \tag{10}$$

F. Security evaluation

Having provided closed-form estimates for the false- and true-detection probabilities in Secs. IV-D–IV-E for our design in Sec. IV-C, we now illustrate the efficacy of this design using empirical data. To do so, there are two more types of information we need.

1) Bounding $\textsc{fdp}(\ell, n)$*:* The first information we need is an estimate of how $\textsc{fdp}(\ell, n)$ should be bounded in practice, as this will permit us to select other parameters of our system to best meet that bound. To find such an estimate, we turned to Florêncio et al. [20], who categorize online guessing attacks into "depth-first" ones that submit many login attempts to few accounts over time and "breadth-first" ones that submit login attempts to many accounts over time (but necessarily fewer per account). Assuming a guessing campaign over a four-month period, they estimate that an account targeted in a depth-first attack should withstand $\ell = 10^6$ guesses, while an account included in a breadth-first attack should withstand $\ell = 10^4$ guesses [20, Table 5]. While a *raat* in our context is not trying to guess the user-chosen password for an account (which it might already know), it is instead trying to guess a honeyword; nevertheless, we take the characterization of online guessing campaigns by Florêncio et al. as suitable for *raats*, as well. Unfortunately, Florêncio et al. did not attempt to precisely characterize with what probability an account subjected to ℓ login attempts should remain uncompromised, nor did they specify the number n of attacked accounts that distinguish breadth-first from depth-first attacks. As such, noticing that $\textsc{fdp}(10^6, 10) \approx \textsc{fdp}(10^4, 1000)$, it is convenient to require

$$\textsc{fdp}(10^6, 10) \le \epsilon \quad \text{and} \quad \textsc{fdp}(10^4, 1000) \le \epsilon \tag{11}$$

for a fixed $\epsilon \le 10^{-1}$ as consistent with the analysis of Florêncio et al. We stress that this requirement is not per login attempt or per account, but *per online-guessing campaign*. If the campaigns envisioned by these authors (each four months long) were mounted consecutively, $\epsilon \le 10^{-1}$ would imply less than one false detection *every 3 years* (= four months per campaign × 9 campaigns) in expectation.

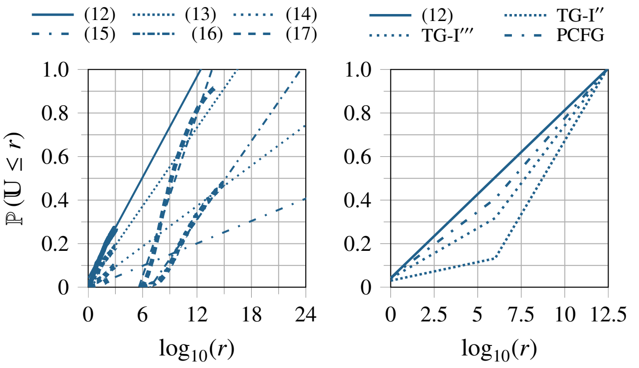
2) Estimating the distribution of $\mathbb{U}$*:* The second type of information we need is the distribution of $\mathbb{U}$, i.e., the random variable that records the rank of the user-chosen password for an account in the list $\mathsf{pwd}[\cdot]$ of passwords ranked in order of the user's likelihood of choosing them, from a *brat*'s perspective. We need this distribution to calculate (2). We use three sources for distributions of $\mathbb{U}$ below.

Wang et al. [40] studied algorithms to guess user passwords from personal information about users. For a breached Chinese train-ticketing dataset of 129,303 passwords with accompanying personal information (name, username, national identification number, phone number, birth date, and email address), they reported the fraction y of accounts they cracked in half the dataset ($N = 64{,}651$ accounts) after training with the other half, as a function the ($\log_{10}$ of the) number x of guesses by the attacker up to $x \le 1000$ [40, Fig. 8]. Due to the unavailability of Wang et al.'s source data or algorithm implementations, we extracted the source data underlying this plot using WebPlotDigitizer [32] for the following algorithms listed there: TarGuess-I (TG-I), which utilizes all of the personal data; TarGuess-I″ (TG-I″), which uses only name and birth date; TarGuess-I‴ (TG-I‴), which uses only name; and PCFG, which leverages no personal information. We fit lines[1] to each dataset, obtaining estimates for $\mathbb{P}\left(\mathbb{U}_{\textsc{tg-i}} \le r\right)$,

[1] Password frequencies are sometimes modeled using a Zipf distribution (e.g., [6], [26], [38]). However, we know of no studies on the effects of attacker knowledge (as we consider here) on this modeling. Moreover, our fitting CDFs achieve better R^2 and RMSE measures on both datasets than our attempts using a Zipf distribution or power regression did.

Data source	Best fit $\mathbb{P}(\mathbb{U} \le r) = \alpha + \beta \log_{10}(r)$		R^2	RMSE	
TG-I [40, Fig. 8]	$\alpha = 0.0419$	$\beta = 0.0771$	0.99	0.004	(12)
TG-I″ [40, Fig. 8]	$\alpha = 0.0118$	$\beta = 0.0601$	0.99	0.004	(13)
TG-I‴ [40, Fig. 8]	$\alpha = 0.0013$	$\beta = 0.0310$	0.97	0.005	(14)
PCFG [40, Fig. 8]	$\alpha = -0.0047$	$\beta = 0.0172$	0.96	0.003	(15)
CMU [28, Fig. 7]	$\alpha = -0.4119$	$\beta = 0.0602$	0.98	0.022	(16)
CKL-PCFG [43, Fig. 5]	$\alpha = -0.6938$	$\beta = 0.1240$	0.97	0.053	(17)

(a) Data sources and best-fit CDFs



(b) Original (thicker) CDFs and best-fit (thinner) CDFs for data sources in Fig. 1a

(c) CDF (12) and CDF (18) with various indicated blocklisting algorithms

Fig. 1: Data sources used in our evaluations

$\mathbb{P}(\mathbb{U}_{\text{TG-I}''} \le r)$, $\mathbb{P}(\mathbb{U}_{\text{TG-I}'''} \le r)$, and $\mathbb{P}(\mathbb{U}_{\text{PCFG}} \le r)$ (Figs. 1a–b).

As discussed in Sec. I, a motivation for Bernoulli honeywords is detecting a *brat* who knows more about users than the defending site does. To show this benefit with these distributions, we allow the *brat* to attack using TG-I (the most user information), but the defending site to blocklist passwords guessable in 10^6 guesses using TG-I″, TG-I‴, or PCFG (i.e., less user information), preventing the user from setting such a password. We select a per-user blocklist of size 10^6 in accordance with blocklist size recommendations (e.g., [34]) and since passwords guessable in 10^6 guesses are typically categorized as *weak* by password strength meters (e.g., [22], [43]). To estimate the effects of this blocklisting, we formulate $\mathbb{P}(\mathbb{U} \le r)$ using two line segments, one for $r \le 10^6$ in which $\mathbb{P}(\mathbb{U} \le r)$ is suppressed by the blocklist, and one for $r > 10^6$ in which the ground lost when $r \le 10^6$ is recovered. That is, for blocklisting algorithm $A \in \{\text{TG-I}'', \text{TG-I}''', \text{PCFG}\}$, we evaluate TDP($n$) using

$$\mathbb{P}(\mathbb{U} \le r) = \begin{cases} \mathbb{P}(\mathbb{U}_{\text{TG-I}} \le r) - \mathbb{P}(\mathbb{U}_{A} \le r) & \text{if } 1 \le r \le 10^6 \\ \left(\frac{1 - \mathbb{P}(\mathbb{U} \le 10^6)}{\log_{10}(r^*) - 6}\right)(\log_{10}(r) - 6) & \text{if } 10^6 < r \le r^* \\ \quad + \ \mathbb{P}\left(\mathbb{U} \le 10^6\right) & \end{cases} \tag{18}$$

where r^* is the minimum r satisfying $\mathbb{P}(\mathbb{U}_{\text{TG-I}} \le r) = 1$ (see Fig. 1c). We concede that this estimate is likely very rough, but we know of no better way to estimate the effects of blocklisting on subsequent password choices, i.e., once one's initial choice has been declined and the user has been told to avoid including personal information in her password.

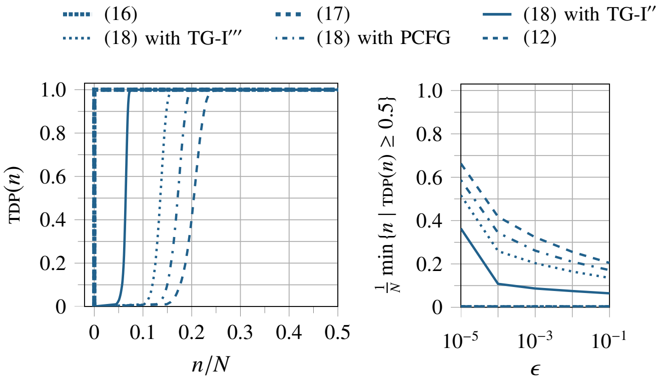
The second source from which we estimate $\mathbb{P}(\mathbb{U} \le r)$ is Mazurek et al. [28], who studied > 25,000 passwords in use at Carnegie Mellon University (CMU). They analyzed these passwords' guessing resistance up to 3.8×10^{14} guesses by an "extensive knowledge" attacker trained on a subset of the passwords in use. This attacker was thus partially trained on passwords that presumably reflected an affiliation at CMU, which constitutes a type of personal information. Fitting a line to points extracted by WebPlotDigitizer from the CDF for the guessing success of this attacker against $N = 5{,}459$ accounts [28, Fig. 7], we obtained an estimate for $\mathbb{P}(\mathbb{U} \le r)$, also shown in Figs. 1a–b.

We get a third estimate for $\mathbb{P}(\mathbb{U} \le r)$ from Xu et al. [43], who propose password guessing models based on "chunk-level" password characteristics and show that their models' guessing accuracies outperform counterparts working at character-level granularity. For example, their "CKL-PCFG" model guessed on average 51.2% more passwords than state-of-the-art PCFG models, including the PCFG model used by Wang et al. [40]. Their evaluation included a "Neopets" dataset of $N = 67{,}672{,}205$ account-password pairs breached from a virtual pets website. Since they trained their password-guessing algorithm using some of the passwords in the Neopets dataset, it implicitly incorporates some private information (e.g., interest in virtual pets) about the site's users. Again, we estimate $\mathbb{P}(\mathbb{U} \le r)$ by fitting a line to data points extracted by WebPlotDigitizer from their subfigure labeled "CKL-PCFG" [43, Fig. 5] (Figs. 1a–b). We did not explore blocklisting in tests using the CMU or CKL-PCFG estimates, though since these passwords were much stronger than the passwords studied by Wang et al. [40], we will see that they nevertheless yield far higher estimates for TDP(n).

3) Results: Note that TDP(n) is itself a random variable, since it depends on the rank sets $\mathcal{R}_a$ selected per account a. Computing statistics of the distribution of TDP(n) is costly, however, since it involves summing over values for the set $\mathcal{R} = \{r_j\}_{j=1}^{\theta}$ and accumulating their probabilities per (1), with each r_j ranging beyond 10^{30} for some of our datasets in Sec. IV-F2. For this reason, here we simulate results by sampling $\mathcal{R}_a$ for each account a, for $\theta = 1000$. We do this for each of the datasets described in Sec. IV-F2—i.e., sampling $\mathcal{R}_a$ for each of $N = 64{,}651$, $N = 5{,}459$, and $N = 67{,}672{,}205$ accounts in the three datasets—and then simulate the *brat* attacking these accounts in increasing order of (2) (and so (8)). We repeat this sample-then-attack experiment for 10,000 trials and report the fraction in which detection occurred within the first n accounts attempted as TDP(n).

We show the results in Fig. 2. These curves were plotted by setting the Bloom-filter dimensions to $k = 20$ and $b = 128$ and then setting p_{h} so to ensure (11). We chose k and b to get adequate granularity of datapoints for plotting these curves. In practice, smaller values b and k, e.g., $b = 64$ or/and $k = 10$, could ensure roughly the same accuracy.

Fig. 2a shows true-detection probabilities using the distributions shown in Fig. 1, as a function of the fraction n/N of accounts accessed by the *brat*, when $\epsilon = 10^{-1}$. This figure shows that using (12) for $\mathbb{P}(\mathbb{U} \le r)$, the true-detection probability reaches 0.5 when the *brat* accesses ≈ 20.4% of the accounts, and it reaches 1.0 when that fraction reaches



(a) $\epsilon = 10^{-1}$

(b) $\textsc{tdp}(n) \geq 0.5$

Fig. 2: True detection probability for honeychecker integration (Sec. IV-C). Fig. 2a shows $\textsc{tdp}(n)$ per fraction n/N of accounts accessed by the *brat*. Fig. 2b shows the fraction n/N of accounts attempted by the *brat* at which $\textsc{tdp}(n) \geq 0.5$, per bound ϵ in (11). TG-I″, TG-I‴, and PCFG represent different blocklists adopted for (18); see Sec. IV-F2.

≈ 26.1%. We reiterate that since the *brat* accesses accounts in increasing order of (2), this curve represents the best the *brat* can do to evade detection subject to the number of accounts he accesses. The total number of accessed accounts to make detection likely, however, is somewhat large, since these passwords are quite weak and so easily predictable by the *brat*. For this reason, blocklisting helps considerably: e.g., when blocklisting (i.e., using (18)) with TG-I‴, the true-detection probability passes 0.5 at only ≈ 13.6%, and with TG-I″, the probability surpasses 0.5 at ≈ 6.4%. The results using (16) or (17) for $\mathbb{P}(\mathbb{U} \leq r)$ are stronger still: after the *brat* accesses only one account, the true-detection probability is already ≈ 1.0. That is, due to the strength of these datasets, even the accounts with the smallest (2) have rank-sets with the lowest-ranked password being a honeyword with near certainty.

Fig. 2b shows the impact of constraining the false-positive probability even more stringently than $\epsilon = 10^{-1}$. This graph shows that driving ϵ lower decreases the true-detection probability in some cases. More specifically, when $\epsilon = 10^{-5}$ and for the weakest password distribution we consider (12), a *brat* can access more than 65% of the accounts before the breach is detected with probability ≥ 0.5. That said, such a stringent ϵ implies less than one false detection per several millennia, in expectation (see Sec. IV-F1). The strongest datasets ((16), (17)) withstand even such a stringent ϵ with no impact on true-detection probability.

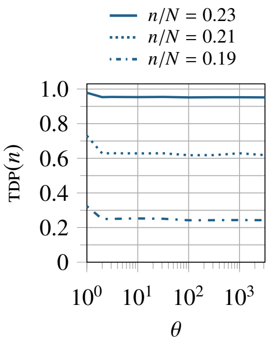


Fig. 3: True-detection probability for honeychecker integration (Sec. IV-C) as function of θ for varying fractions n/N of accounts accessed by the *brat*, based on (12); see Sec. IV-F2.

Our true-detection results in Fig. 2 were computed using $\theta = 1000$, i.e., assuming the *brat* had determined the 1000 lowest-ranked passwords present in the Bloom filter for each account. A natural question is whether setting $\theta = 1000$ in our analyses is sufficiently conservative. Fig. 3 confirms that it clearly is. Specifically, this figure demonstrates that true-detection probabilities decrease noticeably when increasing θ from $\theta = 1$ to $\theta = 2$, but increasing it further has essentially no effect on true detections. Intuitively, the absence of an effect for $\theta > 2$ indicates that while the difference between the probabilities of the two lowest-ranked passwords in an account's Bloom filter—i.e., how "isolated" the first-ranked password is—provides guidance for which account to attack first, additional passwords provide the *brat* little additional information.

V. Integration with Amnesia

The Amnesia design [42] improves on that of Juels & Rivest [25] in two ways. First, it eliminates the assumption that honeychecker state remains secret past the breach of a target site; indeed, the target site in Amnesia has no honeychecker at all. Second, it enables a target to request that another site monitor for the entry of its honeywords, without disclosing them (or the user-chosen password) to the monitoring site and without exposing login attempts at the monitoring site to the target site unless the login attempt actually involves one of the target's honeywords. This remote monitoring is important since a *brat* can distinguish a user-chosen password from honeywords by stuffing them at other sites; since users often reuse a password across sites [13], [30], [37], [27], the user-chosen password at the target emerges as the one that works elsewhere.

In this section we describe an adaptation of the Amnesia framework using the insights of Sec. III. As we will see, this adaptation does come with some consequences in terms of security against *brats*, which we will detail. This section will also leverage Bloom filters, as presented in Sec. IV-A.

A. Detecting a breach locally

1) Background on local detection in Amnesia: In Amnesia, the site forgoes a honeychecker and indeed does not have any ability to distinguish the user-chosen password from honeywords for an account. So, to detect a breach using honeywords, the site needs some other way to determine that the account has been accessed using a honeyword. Amnesia does so by detecting probabilistically if an account has been successfully accessed by two distinct passwords—one of which must be a honeyword. Note, moreover, that Amnesia must do so without storing any state that would indicate to a *brat* what password was previously used to access the account, since that would reveal the user-chosen password to the *brat*, enabling it to avoid using a honeyword.

To achieve this, Amnesia attaches one-bit *marks* to the (hashes of) passwords for an account, so that the password with which the account was last accessed is marked (i.e., its mark is set to 1) and each other password is marked with a certain probability. The user-chosen password is the only one that the intended user should ever use, and so her accesses leave this one marked all the time. If a *brat* accesses the account using a honeyword, however, then the user-chosen password becomes *unmarked* with some probability. In that case, the legitimate user's next login triggers a breach alert, due to using an unmarked password.

Several subtleties in the security of Amnesia against a *brat* were explored in its original analysis [42]. The first is that a *brat* that continues to watch the persistent storage of the breached site as its users log in over time—essentially breaching the site repeatedly across some number L of legitimate logins per account—can narrow in on the user-chosen password as one of those that remain marked across those L logins. To do so, however, the *brat* must remain in the system and exfiltrate this data over time, which presumably leaves the *brat* at greater risk of exposure. Amnesia therefore assumes that L can be reasonably bounded (or that the *brat* will be noticed by other means if not).

The second subtlety in the Amnesia analysis is that once the *brat* decides to access an account using one of the passwords π that remained marked through the L logins, it can do so many times—these logins are indexed $L+1, \ldots, L'$ below—in an attempt to ensure that the *next* most-likely password $\hat{\pi}$, from the *brat*'s perspective, remains marked after login L'. If the *brat* succeeds and if $\hat{\pi}$ is in fact the user-chosen password, then the user entering it will not trigger an alarm. Amnesia therefore additionally requires accounts to be monitored for an unusually high frequency of *successful* logins, e.g., triggering a second-factor or backup authentication challenge if that frequency becomes abnormally large.

Characterizing the effects of L and L' on *brat* detection is challenging. In the Amnesia paper [42], the authors resorted to probabilistic model-checking to analyze these effects and, indeed, did not quantify a true-detection rate for their design. In contrast, our design below supports the first closed-form (albeit approximate) solution for the impact of L and L' on its true-detection probability. We use this solution to show the influence of these parameters on the design.

2) Adapting local detection in Amnesia: In adapting Amnesia to leverage the insights of Sec. III, we adapt a Bloom filter to accommodate marks. Specifically, we subsequently denote a *marked Bloom filter* as a triple $\langle \mathcal{F}, \mathcal{B}, \mathcal{M} \rangle$ where $\mathcal{F}$ and $\mathcal{B}$ are as before and where $\mathcal{M} \subseteq \mathcal{B}$ includes the indices in $\mathcal{B}$ that are marked. Upon a login attempt with password π, the result is determined as follows:

$$\mathsf{login}(\pi) = \begin{cases} \textit{failure} & \text{if } H(\pi) \notin_{\mathrm{B}} \langle \mathcal{F}, \mathcal{B} \rangle \\ \textit{success} & \text{if } H(\pi) \in_{\mathrm{B}} \langle \mathcal{F}, \mathcal{M} \rangle \\ \textit{alarm} & \text{otherwise} \end{cases}$$

If $\mathsf{login}(\pi) = \textit{success}$, a remarking occurs with probability p_{remark}. In a remarking, the set $\mathcal{M}$ is reset to include $\mathcal{F}(H(\pi))$ (i.e., $\mathcal{F}(H(\pi)) \subseteq \mathcal{M}$ with probability 1.0) and each element of $\mathcal{B} \setminus \mathcal{F}(H(\pi))$ independently with probability p_{mark}.

3) Security against a raat: As in Sec. IV, the false-detection probability is simple to characterize for this design. By definition, a *raat* does *not* breach the system and so cannot observe $\langle \mathcal{F}, \mathcal{B}, \mathcal{M} \rangle$ directly for an account. So, when attempting to enter any honeyword in ℓ logins per account, for n accounts, the false-detection probability $\textsc{fdp}(\ell, n)$ can again be bounded as in (7).

4) Security against a brat: The threat model envisioned by Amnesia permits the *brat*, in our case, to capture snapshots of the marked Bloom filter $\langle \mathcal{F}, \mathcal{B}, \mathcal{M} \rangle$ after multiple logins by the legitimate user. If the *brat* breaches the site, capturing the current marked Bloom filter $\langle \mathcal{F}, \mathcal{B}, \mathcal{M}_0 \rangle$, and then continues to monitor the site while the user successfully logs in L times, then the *brat* observes consecutive marked Bloom filters $\{\langle \mathcal{F}, \mathcal{B}, \mathcal{M}_l \rangle\}_{l=0}^{L}$, and a password remains viable only if it is contained in all of them. That is, in the terminology of Sec. III, $s[r] = 1$ iff $\bigwedge_{l=0}^{L} (H(\mathsf{pwd}[r]) \in_{\mathrm{B}} \langle \mathcal{F}, \mathcal{M}_l \rangle)$. Since for $e \overset{\$}{\leftarrow} \{0,1\}^{\nu}$,

$$\mathbb{P}\left(e \in_{\mathrm{B}} \langle \mathcal{F}, \mathcal{M}_0 \rangle \,\middle|\, e \in_{\mathrm{B}} \langle \mathcal{F}, \mathcal{B} \rangle\right) \approx (p_{\mathsf{mark}})^{\mathbb{E}(|\mathcal{F}(e)|)}$$

and for any $l \geq 0$,

$$\begin{aligned} &\mathbb{P}\left(e \in_{\mathrm{B}} \langle \mathcal{F}, \mathcal{M}_{l+1} \rangle \,\middle|\, e \in_{\mathrm{B}} \langle \mathcal{F}, \mathcal{M}_l \rangle\right) \\ &\quad \approx (1 - p_{\mathsf{remark}}) + p_{\mathsf{remark}} (p_{\mathsf{mark}})^{\mathbb{E}(|\mathcal{F}(e)|)} \end{aligned}$$

it is prudent to measure $\textsc{hwin}^{brat}(\cdot,\cdot)$ from Sec. III-B using

$$\begin{aligned} \hat{p}_{\mathsf{h}}(L) = {} & p_{\mathsf{h}} \times (p_{\mathsf{mark}})^{\mathbb{E}(|\mathcal{F}(e)|)} \times \\ & \left(1 - p_{\mathsf{remark}} + p_{\mathsf{remark}} (p_{\mathsf{mark}})^{\mathbb{E}(|\mathcal{F}(e)|)}\right)^{L} \end{aligned} \tag{19}$$

as its first argument. ($\mathbb{E}(|\mathcal{F}(e)|)$ can be instantiated using (4).) As discussed above, this degradation in the probability with which a *brat* inputs a honeyword (i.e., reflected in our use of $\hat{p}_{\mathsf{h}}(L)$ in lieu of p_{h} in $\textsc{hwin}^{brat}(\hat{p}_{\mathsf{h}}(L), \mathcal{R})$) is not an artifact of our construction, but rather an analogous degradation is present in the original Amnesia design [42].

The entry of a honeyword π by a *brat* is necessary but not sufficient to detect the *brat*; in addition, its doing so (after observing L logins by the legitimate user, and then himself logging in another $L' - L$ times) must leave $H(\hat{\pi}) \notin_{\mathrm{B}} \langle \mathcal{F}, \mathcal{M}_{L'} \rangle$, for the user-chosen password $\hat{\pi}$. Recall that by the analysis of Sec. III-B, the *brat* can do no better than attempting the most likely password in its rank ordering $\mathsf{pwd}[\cdot]$ that its monitoring suggests is still viable. To provide a conservative analysis, suppose that the next most-likely viable password $\hat{\pi}$ is indeed the only other possibility for the user-chosen password, in the *brat*'s view. Denote $\hat{\mathcal{S}} = \mathcal{F}(H(\hat{\pi}))$ and $\mathcal{S} = \mathcal{F}(H(\pi))$. Then,

$$\mathbb{P}\left(H(\hat{\pi}) \notin_{\mathrm{B}} \langle \mathcal{F}, \mathcal{M}_{L+1} \rangle\right) \approx p_{\mathsf{remark}} \left(1 - (p_{\mathsf{mark}})^{\mathbb{E}(|\hat{\mathcal{S}} \setminus \mathcal{S}|)}\right) \tag{20}$$

where $\mathbb{E}\left(\left|\hat{\mathcal{S}} \setminus \mathcal{S}\right|\right)$ can be evaluated as in (6), and for $l > L$,

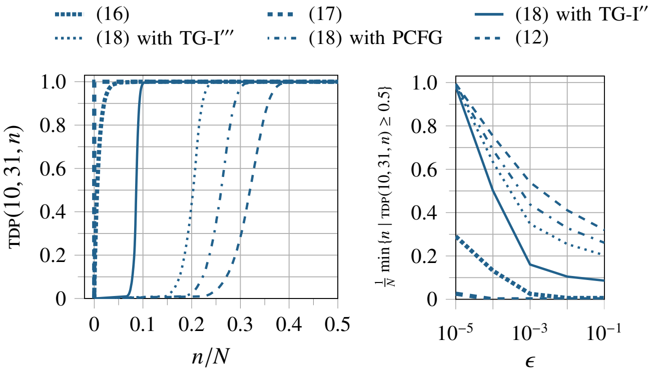
$$\begin{aligned} &\mathbb{P}\left(H(\hat{\pi}) \notin_{\mathrm{B}} \langle \mathcal{F}, \mathcal{M}_{l+1} \rangle \,\middle|\, H(\hat{\pi}) \notin_{\mathrm{B}} \langle \mathcal{F}, \mathcal{M}_l \rangle\right) \\ &\quad \approx (1 - p_{\mathsf{remark}}) + p_{\mathsf{remark}} \left(1 - (p_{\mathsf{mark}})^{\mathbb{E}(|\hat{\mathcal{S}} \setminus \mathcal{S}|)}\right) \end{aligned} \tag{21}$$

The true-detection probability for a *brat* who attacks a $\mathcal{R}$-account, then, is

$$\begin{aligned} \textsc{tdp}_{\mathcal{R}}(L, L') \approx {} & \textsc{hwin}^{brat}(\hat{p}_{\mathsf{h}}(L), \mathcal{R}) \times \mathbb{P}\left(H(\hat{\pi}) \notin_{\mathrm{B}} \langle \mathcal{F}, \mathcal{M}_{L+1} \rangle\right) \\ & \times \prod_{l=L+1}^{L'-1} \mathbb{P}\left(H(\hat{\pi}) \notin_{\mathrm{B}} \langle \mathcal{F}, \mathcal{M}_{l+1} \rangle \,\middle|\, H(\hat{\pi}) \notin_{\mathrm{B}} \langle \mathcal{F}, \mathcal{M}_l \rangle\right) \end{aligned}$$

for any $L' \geq L$. The factors can be plugged in from (19), (20), and (21). Similar to Sec. IV-E, the true-detection probability for a *brat* who attacks accounts $\mathcal{A}' \subseteq \mathcal{A}$ is

$$\textsc{tdp}(L, L', \mathcal{A}') = 1 - \prod_{a \in \mathcal{A}'} (1 - \textsc{tdp}_{\mathcal{R}_a}(L, L')) \tag{22}$$



(a) $\epsilon = 10^{-1}$

(b) $\text{TDP}(10, 31, n) \geq 0.5$

Fig. 4: True detection probability for Amnesia integration (Sec. V-A2). Fig. 4a shows $\text{TDP}(10, 31, n)$ per fraction n/N of accounts accessed by the *brat*. Fig. 4b shows the fraction n/N of accounts attempted by the *brat* at which $\text{TDP}(10, 31, n) \geq 0.5$, per bound ϵ in (11). TG-I″, TG-I‴, and PCFG represent different blocklists adopted for (18); see Sec. IV-F2.

and the *minimum* true-detection probability for *brat* who attacks n accounts is

$$\text{TDP}(L, L', n) = \min_{\mathcal{A}' \subseteq \mathcal{A} : |\mathcal{A}'| = n} \text{TDP}(L, L', \mathcal{A}') \tag{23}$$

5) Security evaluation: We now provide results for the Amnesia integration given in Sec. V-A2, analogous to those of Sec. IV-F but informed by the analysis in Secs. V-A3–V-A4. We again take the requirement (11) on FDP and estimate $\mathbb{P}(\mathbb{U} \leq r)$ using (12), (16), (17), or (18) with some blocklisting algorithm.

Analogous to Fig. 2a, in Fig. 4a we plot $\text{TDP}(L, L', n)$ as a function of the fraction n/N of accounts accessed by the *brat*, produced using 10,000 simulations with $\theta = 1000$ and $\epsilon = 10^{-1}$. For these plots, we set $L = 10$ and $L' = 31$ (i.e., the *brat* logs in up to 20 additional times after its first); we will illustrate the effects of varying these parameters below. We again set $k = 20$ and $b = 128$ and chose $p_{\mathsf{mark}} = 0.95$ and $p_{\mathsf{remark}} = 0.065$. Fig. 4a again highlights the utility of blocklisting; e.g., blocklisting using TG-I″ reduces the fraction of accounts accessed by the *brat* at which the true-detection probability reaches 0.5 from ≈ 32% with no blocklisting to ≈ 8.6%. At the same time, comparing Fig. 4a with Fig. 2a, we see that the *brat* can access ≈ 34.4% more accounts than in the honeychecker design of Sec. IV-C before the true-detection probability reaches this value (in the case of blocklisting with TG-I″), at least for $L = 10$ and $L' = 31$. As such, it is evident that the feature that makes Amnesia symmetric in the sense of Sec. II, i.e., that the site has no data for which its secrecy survives the breach, comes at a cost in detection power. Also, Fig. 4a again highlights the relative strengths of the CMU and CKL-PCFC datasets, showing that our Amnesia integration will detect a *brat* with certainty after it accesses only about 5% and 0.0004% of the accounts in these datasets, respectively.

Fig. 4b shows the impact of constraining the false-positive probability to be $\epsilon \leq 10^{-1}$. Comparing to Fig. 2b, the cost of Amnesia's weak assumptions is evident, in that when $\epsilon = 10^{-5}$, there is minimal true-detection power for a dataset as weak as (12), even with blocklisting. However, datasets (16) and (17) retain substantial true-detection power even at $\epsilon = 10^{-5}$, in that the *brat* will be detected with probability ≥ 0.5 after it accesses ≈ 30% of the accounts.

L	$L' - L$				
	1	6	11	16	21
0	.998	.995	.990	.981	.967
5	.997	.995	.989	.980	.967
10	.997	.994	.989	.980	.966
15	.997	.994	.989	.979	.965
20	.997	.994	.988	.977	.965

(a) $n/5{,}459 \approx 0.03$

L	$L' - L$				
	1	6	11	16	21
0	1.00	1.00	1.00	.999	.997
5	1.00	1.00	1.00	.998	.997
10	1.00	1.00	.999	.998	.997
15	1.00	1.00	.999	.998	.997
20	1.00	1.00	.999	.997	.996

(b) $n/5{,}459 \approx 0.05$

TABLE I: $\text{TDP}(L, L', n)$ with varying L and $L' - L$ based on estimate (16) for $\mathbb{P}(\mathbb{U} \leq r)$

The impact of L and L' are shown in Table I. Recall that L is the number of logins by the legitimate user across which the *brat* monitors the password database before accessing the account himself, and $L' - L$ is the number of logins by the *brat* to first access the account and then to attempt to return the system to a state in which the next login by the legitimate user will not trigger an alarm. Table I suggests the true-detection probability decays modestly when L and $L' - L$ increase.

B. Detecting a breach with remote help

1) Background on remote monitoring in Amnesia: An aspect of the Amnesia design that requires a bit more adaptation to accommodate our approach here is a target site's ability to solicit help from other sites to monitor for the entry of the target's honeywords at those monitoring sites. Critically, Amnesia enables this monitoring without the target T disclosing its honeywords (or the user-chosen password) for accounts to the monitoring site M; without M being able to induce a breach alarm at T with any higher probability than a *raat* could; and without placing M's accounts at risk. Remote monitoring is useful because attempting passwords breached from T at other sites is an effective way to find the user-chosen password, since users tend to reuse passwords across sites [13], [30], [37], [27].

Amnesia achieves remote monitoring via a protocol it calls "private containment retrieval," denoted PCR. To request that M monitor for an account at T, T sends to M a data structure that contains the set of hashes of passwords (one real, and the others honeywords) for that account at T. This data structure is encrypted under a public key *pk* whose private key *sk* is known only to T.

Upon receiving a login attempt for the same user's account at M for which the submitted password π is incorrect (even accounting for typos, e.g., [9]), M inputs $H(\pi)$ as the *test plaintext* and a specific value m as the *response plaintext* to a local *response computation*, together with the encrypted data structure received from T. If the test plaintext $H(\pi)$ matches a hash value in the (plaintext of the) encrypted data structure, then this computation produces a ciphertext of m. Otherwise, it produces a ciphertext of a random plaintext. Even though M knows the ciphertext produced is either of m or of a uniformly random plaintext, M nevertheless cannot tell which type of ciphertext it produced (see [42]).

M returns this ciphertext to T, who decrypts it using sk. T can then test whether the resulting plaintext is the response plaintext m corresponding to any π in its set of passwords for this account. If so, T acts (for the purposes of breach detection) as if π had been input in a local login attempt to this account.

2) Adapting remote monitoring in Amnesia: We describe a way in App. A to use the PCR protocol summarized above to enable remote monitoring for entry of Bernoulli honeywords in the Amnesia integration of Sec. V-A. However, this adaptation costs k PCR responses per incorrect login attempt at the monitoring site M, yielding substantially greater costs. Here we instead provide a more efficient protocol to convey π to T iff $H(\pi) \in_{\mathrm{B}} \langle \mathcal{F}, \mathcal{B} \rangle$. Our design improves on that of App. A by roughly an order of magnitude for the common operations, namely the response generation by M and the response processing by T, as we will show.

Cryptographic primitives: Our protocol builds on an encryption scheme $\mathcal{E}$ with algorithms Gen, Enc, Dec, and $\times_{[\cdot]}$.

- Gen is a randomized algorithm that outputs a public-key/private-key pair $\langle pk, sk \rangle \leftarrow \mathsf{Gen}()$. The value of pk determines a plaintext space that is a cyclic group $(\mathcal{G}, \circ)$ of prime order q, with generator g and identity element $\mathbf{1}$. We assume below that a password π can be encoded as a plaintext in $\mathcal{G}$. For any $m \in \mathcal{G}$, we use m^{-1} to denote the inverse of m and $m^z = m_1 \circ \ldots \circ m_z$ where each $m_j = m$. The value of pk also determines a ciphertext space $C_{pk} = \bigcup_{m \in \mathcal{G}} C_{pk}(m)$, where $C_{pk}(m)$ denotes the ciphertexts for plaintext m. Below our discussion implicitly assumes that $\left|C_{pk}(m)\right| = \left|C_{pk}(m')\right|$ for any $m, m' \in \mathcal{G}$; this is not necessary, but simplifies the discussion below and holds true in our implementation.
- Enc is a randomized algorithm that on input public key pk and a plaintext $m \in \mathcal{G}$, outputs a ciphertext $c \leftarrow \mathsf{Enc}_{pk}(m)$ chosen uniformly at random from $C_{pk}(m)$.
- Dec is a deterministic algorithm that on input a private key sk and ciphertext $c \in C_{pk}$ for the pk corresponding to sk, outputs the plaintext $m \leftarrow \mathsf{Dec}_{sk}(c)$ such that $c \in C_{pk}(m)$. If $c \notin C_{pk}$, then $\mathsf{Dec}_{sk}(c)$ returns $\perp$.
- $\times_{[\cdot]}$ is a randomized algorithm that, on input a public key pk and ciphertexts $c_1 \in C_{pk}(m_1)$ and $c_2 \in C_{pk}(m_2)$, outputs a ciphertext $c \leftarrow c_1 \times_{pk} c_2$ chosen uniformly at random from $C_{pk}(m_1 \circ m_2)$.

Given this functionality, it will be convenient to define two additional operators. Below, "$\mathbf{Y} \overset{d}{=} \mathbf{Y}'$" denotes that random variables $\mathbf{Y}$ and $\mathbf{Y}'$ are distributed identically.

- The $\prod_{pk}$ operator denotes repetition of $\times_{pk}$, i.e.,

$$\overset{z}{\underset{j=1}{\textstyle\prod_{pk}}} c_j \overset{d}{=} c_1 \times_{pk} c_2 \times_{pk} \ldots \times_{pk} c_z$$

- The $\$_{pk}$ operator produces a random ciphertext of $\mathbf{1}$ if its argument is a ciphertext of $\mathbf{1}$, and otherwise produces a ciphertext of a random element of $\mathcal{G} \setminus \{\mathbf{1}\}$. Specifically,

$$\$_{pk}(c) \overset{d}{=} \overset{z}{\underset{j=1}{\textstyle\prod_{pk}}} c \quad \text{where } z \overset{\$}{\leftarrow} \mathbb{Z}_q^*$$

```
T(⟨$\mathcal{F}, \mathcal{B}$⟩)                         M(·)

t1. $\langle pk, sk \rangle \leftarrow \mathsf{Gen}()$
t2. $b' \leftarrow |\mathcal{B}|$
t3. $\forall j \in \{1, \ldots, b\}$ :
        $c_j \leftarrow \begin{cases} \mathsf{Enc}_{pk}(g^{-1}) & \text{if } j \notin \mathcal{B} \\ \mathsf{Enc}_{pk}(g) & \text{if } j \in \mathcal{B} \end{cases}$
t4. $\Psi \leftarrow \mathsf{zkpGen}(\langle pk, \{c_j\}_{j=1}^b \rangle)$
t5. save $\langle pk, sk \rangle$

r1.  $\xrightarrow{pk, \mathcal{F}, b', \{c_j\}_{j=1}^b, \Psi}$

                  m1. abort if $\neg\mathsf{zkpVerify}(\langle pk, \{c_j\}_{j=1}^b \rangle, \Psi)$
                  m2. $c \leftarrow \overset{b}{\underset{j=1}{\prod_{pk}}} c_j$
                  m3. $d_0 \leftarrow c \times_{pk} \mathsf{Enc}_{pk}(g^{b-2b'})$
                  m4. save $\langle pk, \mathcal{F}, \{c_j\}_{j=1}^b, d_0 \rangle$
```

Fig. 5: Monitor deployment

Protocol description: The protocol for T to deploy a monitor for a selected account to site M is shown in Fig. 5, and the protocol for M to send a monitoring response (upon an attempted login to that account) to T is shown in Fig. 6. Deployment begins by T creating a public/private key pair $\langle pk, sk \rangle$ (line t1) that it will save for processing monitoring responses later (t5). The monitoring request itself includes b ciphertexts $\{c_j\}_{j=1}^b$ that encode which indices j are in the account's Bloom filter indices $\mathcal{B}$ (encoded as $c_j \in C_{pk}(g)$) and which are not (encoded as $c_j \in C_{pk}(g^{-1})$); see line t3. In addition, the monitoring request (message r1) includes pk; the uniform hash functions $\mathcal{F}$ for the account's Bloom filter; the number b' of indices in $\mathcal{B}$ (line t2); and a noninteractive zero-knowledge proof Ψ that $\{c_j\}_{j=1}^b \subseteq C_{pk}(g) \cup C_{pk}(g^{-1})$ (generated using zkpGen in line t4).[2]

Upon receiving a well-formed monitoring request r1 (in particular, where pk is a valid public key), M checks the zero-knowledge proof Ψ using zkpVerify (line m1) and aborts if the check returns *false*. If this check returns *true* and so $\{c_j\}_{j=1}^b \subseteq C_{pk}(g) \cup C_{pk}(g^{-1})$ (except with probability the soundness error of Ψ), then M calculates d_0 to be a ciphertext of $\mathbf{1}$ if and only if the claimed number b' is accurate (lines m2–m3). That is,

$$\begin{aligned} \left|\{c_j\}_{j=1}^b \cap C_{pk}(g)\right| = b' &\Leftrightarrow c \in C_{pk}(g^{b'} g^{-(b-b')}) && \text{in line m2} \\ &\Leftrightarrow d_0 \in C_{pk}(\mathbf{1}) && \text{in line m3} \end{aligned}$$

Finally, M saves pk, $\mathcal{F}$, $\{c_j\}_{j=1}^b$, and d_0 in line m4.

As we will see, $d_0 \notin C_{pk}(\mathbf{1})$ ensures that T learns nothing from M; i.e., T cannot gain any information about logins at M if it reports an incorrect value of b'. b' is reported in r1 primarily to permit M to refuse the monitoring request if b' is larger than M deems appropriate. That is, using b', M can

[2] As in Amnesia, if the password-hashing function H in use at T is salted, then the salt can be sent in r1 to M. Or, H could be implemented as an oblivious pseudorandom function (e.g., [21]) keyed with the salt, which M would evaluate on π with an extra interaction with T in Fig. 6.

```
$\mathsf{T}(\langle \mathcal{F}, \mathcal{B} \rangle, \langle pk, sk \rangle)$        $\mathsf{M}(\langle pk, \mathcal{F}, \{c_j\}_{j=1}^{b}, d_0 \rangle)$

                                  Upon login attempt with incorrect password $\pi$:
                                  m5. $\mathcal{S} \leftarrow \mathcal{F}(H(\pi))$
                                  m6. $d_1 \leftarrow \prod_{pk, j \in \mathcal{S}} \left( c_j \times_{pk} \mathsf{Enc}_{pk}(g^{-1}) \right)$
                                  m7. $\hat{c}_0 \leftarrow \$_{pk}(d_0) \times_{pk} \$_{pk}(d_1)$
                                  m8. $\hat{c}_1 \leftarrow \$_{pk}(\hat{c}_0) \times_{pk} \mathsf{Enc}_{pk}(\pi)$

        r2. $\xleftarrow{\hat{c}_0, \hat{c}_1}$

t6. abort if $\mathsf{Dec}_{sk}(\hat{c}_0) \neq \mathbf{1}$
t7. $\pi \leftarrow \mathsf{Dec}_{sk}(\hat{c}_1)$
t8. abort if $\pi = \perp$ or $H(\pi) \notin_{\mathrm{B}} \langle \mathcal{F}, \mathcal{B} \rangle$
t9. return $\pi$
```

Fig. 6: Monitor response

calculate p_{h} using (3) with $|\mathcal{B}| = b'$ and accept the monitoring request only if p_{h} is acceptably small. If not, M can just drop this request (not shown in Fig. 5).

The monitor site M generates a response using the protocol shown in Fig. 6. The provided (incorrect) password π in a login attempt to the monitored account is used to generate its set $\mathcal{S} \leftarrow \mathcal{F}(H(\pi))$ of Bloom-filter indices, in line m5. For each such index j, the ciphertext $c_j \times_{pk} \mathsf{Enc}_{pk}(g^{-1})$ is calculated in line m6, yielding a ciphertext of $\mathbf{1}$ if $c_j \in C_{pk}(g)$ and a ciphertext of g^{-2} otherwise. These ciphertexts are combined using $\times_{pk}$, ensuring that d_1 is a ciphertext of $\mathbf{1}$ if and only if all of them are. The ciphertext $\hat{c}_0$ is then set to be a ciphertext of $\mathbf{1}$ if and only if both d_0 and d_1 are (with overwhelming probability; line m7), and then $\hat{c}_1$ is set to be a ciphertext of π in that case (and only that case; line m8).

Upon receiving $\hat{c}_0$, $\hat{c}_1$ in message r2, T tests whether $\hat{c}_0 \in C_{pk}(\mathbf{1})$ and aborts if not (line t6); aborting here indicates that $H(\pi) \notin_{\mathrm{B}} \langle \mathcal{F}, \mathcal{B} \rangle$. As such, if the protocol does not abort in line t6 but $H(\pi) \notin_{\mathrm{B}} \langle \mathcal{F}, \mathcal{B} \rangle$ in line t8, then this reveals that M has misbehaved (and so the protocol again aborts). Otherwise, π is returned and T treats it as if it were entered in a local login attempt for this account, for the purposes of breach detection.

We prove the cryptographic security of this protocol in App. B. Below we sketch the cryptographic arguments and additionally discuss other factors that bear on its security for our purposes.

3) Security against a malicious M*:* The first threat that we address relative to a malicious M is the possibility that M learns information about the honeywords or user-chosen password for an account at T for which it is asked to monitor. Of the values M receives in the protocol (see message r1), pk and $\mathcal{F}$ are chosen independently from these passwords; $\{c_j\}_{j=1}^{b}$ are ciphertexts encrypted using pk; and Ψ is a zero-knowledge proof that $\{c_j\}_{j=1}^{b}$ are well-formed. This protocol does disclose $|\mathcal{B}| = b'$ to M, though b' is a tunable parameter chosen by T even prior to user password registration and, combined with $k = |\mathcal{F}|$, merely reveals to M the value of p_{h} (i.e., $p_{\mathsf{h}} = (b'/b)^k$) in use at T. It does not, however, provide the adversary any ability to distinguish $\mathsf{T}(\langle \mathcal{F}, \mathcal{B} \rangle)$ from $\mathsf{T}(\langle \mathcal{F}, \mathcal{B}' \rangle)$ for distinct $\mathcal{B}$ and $\mathcal{B}'$, provided that $|\mathcal{B}| = |\mathcal{B}'| = b'$. Thus, if a malicious M has a non-negligible advantage in distinguishing between $\mathsf{T}(\langle \mathcal{F}, \mathcal{B} \rangle)$ and $\mathsf{T}(\langle \mathcal{F}, \mathcal{B}' \rangle)$ on the basis of message r1, even for $\mathcal{B}$ and $\mathcal{B}'$ of M's own choosing (satisfying $|\mathcal{B}| = |\mathcal{B}'| = b'$), then there is an IND-CPA [3] adversary with non-negligible advantage against $\mathcal{E}$. This claim is in the random oracle model [4]—in particular, the non-interactive proof Ψ built using the Fiat-Shamir heuristic [19] implies random oracles. Below we will instantiate the encryption scheme $\mathcal{E}$ and zero-knowledge proof Ψ in our implementation.

Given that the protocol does not leak information about the honeywords to M, we now consider the threat of M using its vantage point to induce a false breach alarm at T. Interestingly, because M learns $\mathcal{F}$ and might know the user-chosen password $\hat{\pi}$ at T (e.g., if it was reused at M), M *does* gain an advantage in inducing a false alarm at T. Specifically, while M does not know $\mathcal{M}$ or $\mathcal{B}$ at T, it knows that $|\mathcal{B}| = b'$ and $\mathcal{F}(H(\hat{\pi})) \subseteq \mathcal{M}$. So, to attempt to induce a false alarm at T, it can select a password π to maximize

$$\mathbb{P}\left(\begin{matrix} \mathcal{F}(H(\pi)) \subseteq \mathcal{B} \ \wedge \\ \mathcal{F}(H(\pi)) \cap (\mathcal{B} \setminus \mathcal{M}) \neq \emptyset \end{matrix} \ \middle| \ \begin{matrix} |\mathcal{B}| = b' \ \wedge \\ \mathcal{F}(H(\hat{\pi})) \subseteq \mathcal{M} \end{matrix} \right)$$

where the probability is with respect to choice of $\mathcal{B}$ and $\mathcal{M}$ by T. Returning $\hat{c}_0 \in C_{pk}(\mathbf{1})$ and $\hat{c}_1 \in C_{pk}(\pi)$ will then induce a false alarm with higher probability than our calculations in previous sections would suggest. Since we model H as a random oracle, M must search for such a π in a brute-force manner, but it can do so *offline*, whereas a *raat* as discussed previously can attempt to induce a false alarm only as an online attack. Note that this attack is equally relevant for our adaptation of the Amnesia protocol discussed in App. A, since M obtains $\mathcal{F}$ there, as well.

To defend against this added risk of false alarms, T can use two distinct Bloom filters per account for (both local and remote) breach detection: one private Bloom filter $\langle \mathcal{F}_{\mathsf{pr}}, \mathcal{B}_{\mathsf{pr}}, \mathcal{M}_{\mathsf{pr}} \rangle$ with marks and one "public" one $\langle \mathcal{F}_{\mathsf{pu}}, \mathcal{B}_{\mathsf{pu}} \rangle$ without marks. The detection rule in Sec. V-A2 can be modified to be

$$\mathsf{login}(\pi) = \begin{cases} \textit{failure} \text{ if } H(\pi) \notin_{\mathrm{B}} \langle \mathcal{F}_{\mathsf{pr}}, \mathcal{B}_{\mathsf{pr}} \rangle \vee H(\pi) \notin_{\mathrm{B}} \langle \mathcal{F}_{\mathsf{pu}}, \mathcal{B}_{\mathsf{pu}} \rangle \\ \textit{success} \text{ if } H(\pi) \in_{\mathrm{B}} \langle \mathcal{F}_{\mathsf{pr}}, \mathcal{M}_{\mathsf{pr}} \rangle \wedge H(\pi) \in_{\mathrm{B}} \langle \mathcal{F}_{\mathsf{pu}}, \mathcal{B}_{\mathsf{pu}} \rangle \\ \textit{alarm} \quad \text{otherwise} \end{cases}$$

Here, $\mathcal{F}_{\mathsf{pr}}$ and $\mathcal{F}_{\mathsf{pu}}$ are independently chosen. Then, T can use $\langle \mathcal{F}_{\mathsf{pu}}, \mathcal{B}_{\mathsf{pu}} \rangle$ in the protocol of Sec. V-B2 while keeping $\langle \mathcal{F}_{\mathsf{pr}}, \mathcal{B}_{\mathsf{pr}}, \mathcal{M}_{\mathsf{pr}} \rangle$ private. To limit the false detection probability against M to at most that provided in Sec. V-A3 against a local *raat* without knowledge of $\langle \mathcal{F}, \mathcal{B}, \mathcal{M} \rangle$, T can configure $\langle \mathcal{F}_{\mathsf{pr}}, \mathcal{B}_{\mathsf{pr}}, \mathcal{M}_{\mathsf{pr}} \rangle$ as prescribed previously. In this case, even with knowledge of $\mathcal{F}_{\mathsf{pu}}$ and $\hat{\pi}$, M can do no better in triggering a false alarm than a local *raat* could have, due to the privacy of $\langle \mathcal{F}_{\mathsf{pr}}, \mathcal{B}_{\mathsf{pr}}, \mathcal{M}_{\mathsf{pr}} \rangle$ at the *unbreached* T.

In this design, increasing $\left|\mathcal{F}_{\mathsf{pu}}\right|$ reduces the true detection probability. Decreasing $\left|\mathcal{F}_{\mathsf{pu}}\right|$, on the other hand, increases the number of failed login passwords at M that are transmitted to T via the protocol of Sec. V-B2; as such, M might refuse a monitoring request for which $\left|\mathcal{F}_{\mathsf{pu}}\right|$ is too small. Such impacts can be quantified using the principles we developed previously.

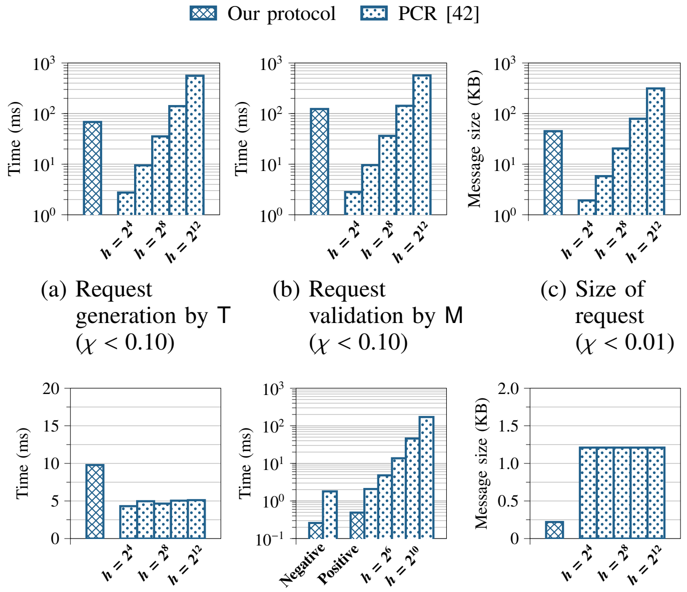
4) Security against a malicious T*:* The security risk that a malicious T poses to a monitor M is that T will learn more about the passwords entered in login attempts at M than it should. The login passwords π on which M executes the response protocol (Fig. 6) are at M's discretion. The key property that we show about this protocol is that if M executes the response protocol on a password π, then no information about π is conveyed to T unless the monitoring request is well-formed and $H(\pi) \in_{\mathrm{B}} \langle \mathcal{F}, \mathcal{B}' \rangle$ for the $\mathcal{B}'$ represented by the monitoring request, no matter how T misbehaves. This security argument follows from three facts.

- If a monitor M accepts a monitor request from T (line m4), then $\{c_j\}_{j=1}^{b} \subseteq C_{pk}(g) \cup C_{pk}(g^{-1})$ except with probability the soundness error of Ψ.
- Let $\mathcal{B}' = \{j : c_j \in C_{pk}(g)\}$. If $\{c_j\}_{j=1}^{b} \subseteq C_{pk}(g) \cup C_{pk}(g^{-1})$ but $|\mathcal{B}'| \neq b'$, then $d_0 \in C_{pk} \setminus C_{pk}(\mathbf{1})$ in line m4. Therefore, $\hat{c}_0 \in C_{pk}(\mathbf{1})$ with probability at most $1/(q-1)$ and otherwise is uniformly distributed in $C_{pk} \setminus C_{pk}(\mathbf{1})$. When $\hat{c}_0 \in C_{pk} \setminus C_{pk}(\mathbf{1})$, $\hat{c}_1$ is uniformly distributed in $C_{pk} \setminus C_{pk}(\pi)$ (line m8).
- If $\{c_j\}_{j=1}^{b} \subseteq C_{pk}(g) \cup C_{pk}(g^{-1})$ and $|\mathcal{B}'| = b'$ but in a run of the response protocol (Fig. 6) on π, it is the case that $H(\pi) \notin_{\mathrm{B}} \langle \mathcal{F}, \mathcal{B}' \rangle$, then the value $\hat{c}_0$ returned from M is uniformly distributed in $C_{pk} \setminus C_{pk}(\mathbf{1})$, and so $\hat{c}_1$ is uniformly distributed in $C_{pk} \setminus C_{pk}(\pi)$. This is immediate since in line m6, d_1 will be generated in $C_{pk}(g^{-2|\mathcal{F}(H(\pi)) \setminus \mathcal{B}'|})$. Therefore, $\hat{c}_0$ is uniformly distributed in $C_{pk} \setminus C_{pk}(\mathbf{1})$ (m7).

Given these facts, M conveys information to T in a run of the response protocol (Fig. 6) only if $\{c_j\}_{j=1}^{b} \subseteq C_{pk}(g) \cup C_{pk}(g^{-1})$, $|\mathcal{B}'| = b'$, and $H(\pi) \in_{\mathrm{B}} \langle \mathcal{F}, \mathcal{B}' \rangle$.

5) Performance: We implemented the protocol of Sec. V-B2 and compared its performance to Amnesia's PCR protocol, retrieved from https://github.com/k3coby/pcr-go. To facilitate comparison between PCR and ours, we implemented ours in Go (as PCR is) and used the same encryption scheme $\mathcal{E}$, namely ElGamal encryption [16] with $\mathcal{G}$ being the elliptic-curve group secp256r1 [7]. The construction of the zero-knowledge proof Ψ is discussed in App. B-A2. Our implementation can be retrieved from https://github.com/k3coby/bhwmonitoring-go. In our experiments, T and M executed on separate AWS EC2 instances having a single 2.50GHz vCPU running Ubuntu 20.04.4. All datapoints are the means of 50 runs; we report relative standard deviations (χ) in each caption.

Fig. 7 provides a comparison of costs. Note that some vertical axes are log-scale. In Fig. 7, we denote by h the number of explicit honeywords with which PCR's monitoring request is configured, up to $h = 2^{12}$; this value is not excessive, but rather even larger numbers are suggested in the Amnesia design to offset certain threats [42, Sec. 5.4]. "Request generation by T" (Fig. 7a) refers to execution at T preceding the request message (i.e., lines t1–t5 in Fig. 5, and the analogous instructions for PCR); "Request validation by M" (Fig. 7b) refers to execution at M following receipt of a well-formed request message (m1–m4); "Response generation by M" (Fig. 7d) refers to execution at M preceding the response message (m5–m8 in Fig. 6); and "Response processing by T" (Fig. 7e) refers to execution at T following receipt of a well-formed response message (t6–t9). In the last case, "Negative" refers to the case $H(\pi) \notin_{\mathrm{B}} \langle \mathcal{F}, \mathcal{B} \rangle$, and "Positive" refers to the case $H(\pi) \in_{\mathrm{B}} \langle \mathcal{F}, \mathcal{B} \rangle$.



(a) Request generation by T ($\chi < 0.10$)

(b) Request validation by M ($\chi < 0.10$)

(c) Size of request ($\chi < 0.01$)

(d) Response generation by M ($\chi < 0.12$)

(e) Response processing by T ($\chi < 0.60$)

(f) Size of response ($\chi < 0.01$)

Fig. 7: Costs for our protocol (Sec. V-B2) and PCR [42]. One-time costs for T to deploy a monitoring request to M are shown in (a)–(c), and costs for M to return a response to T, once per incorrect login attempt, are shown in (d)–(f). In (e), "negative" and "positive" refer to the cases in our protocol when $H(\pi) \notin_{\mathrm{B}} \langle \mathcal{F}, \mathcal{B} \rangle$ and $H(\pi) \in_{\mathrm{B}} \langle \mathcal{F}, \mathcal{B} \rangle$, respectively; PCR has analogous cases. Numbers shown are for $b = 128$ and $k = 20$ for our protocol. h is the number of explicit honeywords with which PCR's monitoring request was configured.

As shown in Fig. 7, in all measures except for response computation by M (Fig. 7d), our protocol eventually outperforms the original PCR protocol for Amnesia as the number of honeywords is increased. Response generation by M is more expensive by a mere 5ms, which could be eliminated by reducing k from $k = 20$ to $k = 10$. We have used $k = 20$ throughout this paper primarily to produce true-detection curves containing more points and so that are smoother. Reducing to $k = 10$ would have little practical effect.

Though we arrived at our remote-monitoring protocol through its support for Bernoulli honeywords, it will work with any Bloom filter provided by T. So, it presents an alternative for remote monitoring in an adaptation of Amnesia using a Bloom filter (vs. a Cuckoo filter [18]), even one populated with explicit honeywords.

VI. Discussion

Online password guessing: Our primary threat models considered in this paper, namely *raats* and *brats*, leave one additional threat model to consider: The risk that an attacker *not* knowing the user-chosen password succeeds in an online password guessing attack to access an account—versus to induce a false breach alarm, as a *raat* does *with* knowledge of the user-chosen password. Denote by $\textsc{gp}(\ell)$ the probability with which an online attacker would succeed in guessing the password for a particular account within ℓ tries. Then, Bernoulli honeywords increase his probability of accessing this account to no more than $\textsc{gp}(\ell) + \textsc{fdp}(\ell, 1)$, since entry of either the user-chosen password or a honeyword is necessary (though not sufficient in our specific designs[3]) to access the account. As discussed in Sec. IV-F, Florêncio et al. [20] recommend resisting up to $\ell = 10^6$ online guesses in a prolonged depth-first campaign, which we interpreted to require $\textsc{fdp}(10^6, 10) \leq 10^{-1}$ (and so $\textsc{fdp}(10^6, 1) < 10^{-2}$). Still, if the user-chosen password satisfies $\textsc{gp}(10^6) \ll 10^{-2}$, then it is conceivable that Bernoulli honeywords could theoretically weaken the account to access by an online guessing attack. That said, any such weakening is of little practical importance, for online guessing attacks or others. Florêncio et al. put it this way: "... consider two passwords which withstand 10^6 and 10^{12} guesses respectively ... there is no apparent scenario in which the extra guess-resistance of the second password helps. ... both will survive online guessing, but neither will survive offline attack" [20, p. 43]. We therefore conclude that the advantages brought by Bernoulli honeywords far outweigh any additional risk of account access by an online guessing attack. In those rare cases of a user-chosen password capable of withstanding even an offline attack, a site can simply exempt this account from using Bernoulli honeywords.

Space efficiency: Though not our primary motivation for using them, Bloom filters are very space-efficient data structures, which has additional benefits for our designs. For example, Wang et al. [41, Sec. V.B] estimates the honeyword storage for 10^7 accounts costs 12.8GB for 40 honeywords per user. In our design, storing Bloom filters with $b = 128$ (the parameter we chose for our security evaluations) for the same number of users would cost 160MB only.

This space efficiency also has benefits for remote monitoring. For example, with $b = 128$ our protocol in Sec. V-B2 would require M to store < 1GB for 10^5 monitoring requests. In contrast, the authors of Amnesia show that if 4096 *explicit* honeywords are deployed for each account, it requires 32GB to store the same number of monitoring requests.

A direction for future research is more space-efficient methods for implementing Bernoulli honeywords. Certain natural candidates, e.g., simply storing $H(\hat{\pi}) \bmod w$ for the user-chosen password $\hat{\pi}$, would implement $p_{\mathsf{h}} = 1/w$. However, a similarly efficient construction is possible using Bloom filters, by setting $b = w$, $k = 1$, and $|\mathcal{B}| = 1$.

VII. Conclusion

In this paper we have made the case for choosing honeywords from all possible passwords as a Bernoulli process, in contrast to previous proposals to generate a small number of honeywords per account using heuristics. We have shown how to realize this idea within existing honeyword system designs, namely the original honeychecker-based design of Juels & Rivest and the more recent Amnesia proposal. Moreover, we have shown that our design enables even greater efficiency than the previous Amnesia proposal for remotely monitoring for the entry of a site's honeywords elsewhere. Most critically, though, we have shown that Bernoulli honeywords permit analytic estimation of true and false breach-detection rates, which we provide for our realizations. In particular, when evaluated against realistic threats, Bernoulli honeywords enable detection of credential database breaches with a risk of false alarms that is quantifiable, tunable, and independent of the adversary's knowledge of the site's users.

Acknowledgment: This research was supported by grant W911NF-17-1-0370 from the U.S. Army Research Office.

References

[1] Akshima, D. Chang, A. Goel, S. Mishra, and S. K. Sanadhya, "Generation of secure and reliable honeywords, preventing false detection," *IEEE Transactions on Dependable and Secure Computing*, vol. 16, no. 5, 2019.

[2] M. H. Almeshekah, C. N. Gutierrez, M. J. Atallah, and E. H. Spafford, "ErsatzPasswords: Ending password cracking and detecting password leakage," in *31st Annual Computer Security Applications Conference*, Dec. 2015.

[3] M. Bellare, A. Desai, D. Pointcheval, and P. Rogaway, "Relations among notions of security for public-key encryption schemes," in *Advances in Cryptology – CRYPTO 1998*, ser. LNCS, vol. 1462, Aug. 1998.

[4] M. Bellare and P. Rogaway, "Random oracles are practical: A paradigm for designing efficient protocols," in *1st ACM Conference on Computer and Communications Security*, Nov. 1993.

[5] B. H. Bloom, "Space/time trade-offs in hash coding with allowable errors," *Communications of the ACM*, vol. 13, no. 7, Jul. 1970.

[6] J. Bonneau, "The science of guessing: analyzing an anonymized corpus of 70 million passwords," in *33th IEEE Symposium on Security and Privacy*, May 2012.

[7] Certicom Research, "SEC 2: Recommended elliptic curve domain parameters," http://www.secg.org/SEC2-Ver-1.0.pdf, 2000, standards for Efficient Cryptography.

[8] N. Chakraborty and S. Mondal, "Toward improving storage cost and security features of honeyword based approaches," *Procedia Computer Science*, vol. 93, 2016.

[9] R. Chatterjee, A. Athayle, D. Akhawe, A. Juels, and T. Ristenpart, "pASSWORD tYPOS and how to correct them securely," in *37th IEEE Symposium on Security and Privacy*, May 2016.

[10] D. Chaum and T. P. Pedersen, "Wallet databases with observers," in *Advances in Cryptology – CRYPTO'92*, ser. LNCS, vol. 740, 1993.

[11] R. Cramer, I. Damgård, and B. Schoenmakers, "Proofs of partial knowledge and simplified design of witness hiding protocols," in *Advances in Cryptology – CRYPTO '94*, ser. LNCS, vol. 839, 1994.

[12] A. Culafi, "Mandiant: Compromised Colonial Pipeline password was reused," https://www.techtarget.com/searchsecurity/news/252502216/Mandiant-Compromised-Colonial-Pipeline-password-was-reused, 09 Jun. 2021.

[13] A. Das, J. Bonneau, M. Caesar, N. Borisov, and X. Wang, "The tangled web of password reuse," in *21st ISOC Network and Distributed System Security Symposium*, 2014.

[14] J. DeBlasio, S. Savage, G. M. Voelker, and A. C. Snoeren, "Tripwire: Inferring internet site compromise," in *17th Internet Measurement Conference*, Nov. 2017.

[15] A. Dionysiou and E. Athanasopoulos, "Lethe: Practical data breach detection with zero persistent secret state," in *7th IEEE European Symposium on Security and Privacy*, Jun. 2022.

[16] T. ElGamal, "A public-key cryptosystem and a signature scheme based on discrete logarithms," *IEEE Transactions on Information Theory*, vol. 31, no. 4, 1985.

[17] I. Erguler, "Achieving flatness: Selecting the honeywords from existing user passwords," *IEEE Transactions on Parallel and Distributed Systems*, vol. 13, no. 2, 2016.

[3] Our honeychecker design (Sec. IV-C) also requires $\mathcal{F}(H(\pi)) = \mathcal{F}(H(\hat{\pi}))$ and our Amnesia adaptation (Sec. V-A2) also requires $H(\pi) \in_{\mathrm{B}} \langle \mathcal{F}, \mathcal{M} \rangle$.

[18] B. Fan, D. G. Andersen, M. Kaminsky, and M. D. Mitzenmacher, "Cuckoo filter: Practically better than Bloom," in *10th ACM Conference on Emerging Networking Experiments and Technologies*, 2014.

[19] A. Fiat and A. Shamir, "How to prove yourself: Practical solutions to identification and signature problems," in *Advances in Cryptology – CRYPTO 1986*, ser. LNCS, vol. 263, Aug. 1986.

[20] D. Florêncio, C. Herley, and P. C. van Oorschot, "An administrator's guide to internet password research," in *28th Large Installation System Administration Conference*, Nov. 2014.

[21] M. J. Freedman, Y. Ishai, B. Pinkas, and O. Reingold, "Keyword search and oblivious pseudorandom functions," in *2nd Theory of Cryptography Conference*, ser. LNCS, vol. 3378, Feb. 2005.

[22] M. Golla and M. Dürmuth, "On the accuracy of password strength meters," in *25th ACM Conference on Computer and Communications Security*, Oct. 2018.

[23] IBM Security, "Cost of a data breach report 2021," https://www.ibm.com/security/data-breach, 2021.

[24] S. Ikeda, "Colonial Pipeline hack connected to password leak of 8.4 billion accounts; attackers got in via an old VPN account," https://www.cpomagazine.com/cyber-security/colonial-pipeline-hack-connected-to-password-leak-of-8-4-billion-accounts-attackers-got-in-via-an-old-vpn-account/, 14 Jun. 2021.

[25] A. Juels and R. L. Rivest, "Honeywords: Making password-cracking detectable," in *20th ACM Conference on Computer and Communications Security*, Nov. 2013.

[26] D. Malone and K. Maher, "Investigating the distribution of password choices," in *21st International World Wide Web Conference*, Apr. 2012.

[27] P. Mayer, C. W. Munyendo, M. L. Mazurek, and A. J. Aviv, "Why users (don't) use password managers at a large educational institution," in *31st USENIX Security Symposium*, Aug. 2022.

[28] M. L. Mazurek, S. Komanduri, T. Vidas, L. Bauer, N. Christin, L. F. Cranor, P. G. Kelley, R. Shay, and B. Ur, "Measuring password guessability for an entire university," in *20th ACM Conference on Computer and Communications Security*, Nov. 2013.

[29] M. Mitzenmacher and E. Upfal, *Probability and Computing: Randomization and Probabilistic Techniques in Algorithms and Data Analysis*. Cambridge University Press, 2005.

[30] S. Pearman, J. Thomas, P. E. Naeini, H. Habib, L. Bauer, N. Christin, L. F. Cranor, S. Egelman, and A. Forget, "Let's go in for a closer look: Observing passwords in their natural habitat," in *24th ACM Conference on Computer and Communications Security*, Oct. 2017.

[31] Reuters, "Hackers attack 20 million accounts on Alibaba's Taobao shopping site," https://www.reuters.com/article/us-alibaba-cyber-idUKKCN0VD14X, 04 Feb. 2016.

[32] A. Rohatgi, "WebPlotDigitizer: Version 4.5," https://automeris.io/WebPlotDigitizer, 2021.

[33] Shape Security, "2018 credential spill report," https://info.shapesecurity.com/rs/935-ZAM-778/images/Shape_Credential_Spill_Report_2018.pdf, 2018.

[34] J. Tan, L. Bauer, N. Christin, and L. F. Cranor, "Practical recommendations for stronger, more usable passwords combining minimum-strength, minimum-length, and blocklist requirements," in *27th ACM Conference on Computer and Communications Security*, Nov. 2020.

[35] K. Thomas, F. Li, A. Zand, J. Barrett, J. Ranieri, L. Invernizzi, Y. Markov, O. Comanescu, V. Eranti, A. Moscicki, D. Margolis, V. Paxson, and E. Bursztein, "Data breaches, phishing, or malware? Understanding the risks of stolen credentials," in *24th ACM Conference on Computer and Communications Security*, 2017.

[36] Y. Tsiounis and M. Yung, "On the security of ElGamal based encryption," in *Public Key Cryptography — PKC 1998*, ser. LNCS, vol. 1431, 1998.

[37] C. Wang, S. T. K. Jan, H. Hu, D. Bossart, and G. Wang, "The next domino to fall: Empirical analysis of user passwords across online services," in *8th ACM Conference on Data and Application Security and Privacy*, Mar. 2018.

[38] D. Wang, H. Cheng, P. Wang, X. Huang, and G. Jian, "Zipf's law in passwords," in *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 11, Nov. 2017.

[39] D. Wang, H. Cheng, P. Wang, J. Yan, and X. Huang, "A security analysis of honeywords," in *25th ISOC Network and Distributed System Security Symposium*, Feb. 2018.

[40] D. Wang, Z. Zhang, P. Wang, J. Yan, and X. Huang, "Targeted online password guessing: An underestimated threat," in *23rd ACM Conference on Computer and Communications Security*, 2016.

[41] D. Wang, Y. Zou, Q. Dong, Y. Song, and X. Huang, "How to attack and generate honeywords," in *43rd IEEE Symposium on Security and Privacy*, May 2022.

[42] K. C. Wang and M. K. Reiter, "Using Amnesia to detect credential database breaches," in *30th USENIX Security Symposium*, Aug. 2021.

[43] M. Xu, C. Wang, J. Yu, J. Zhang, K. Zhang, and W. Han, "Chunk-level password guessing: Towards modeling refined password composition representations," in *28th ACM Conference on Computer and Communications Security*, Nov. 2021.

Appendix A
A Remote-Monitoring Alternative

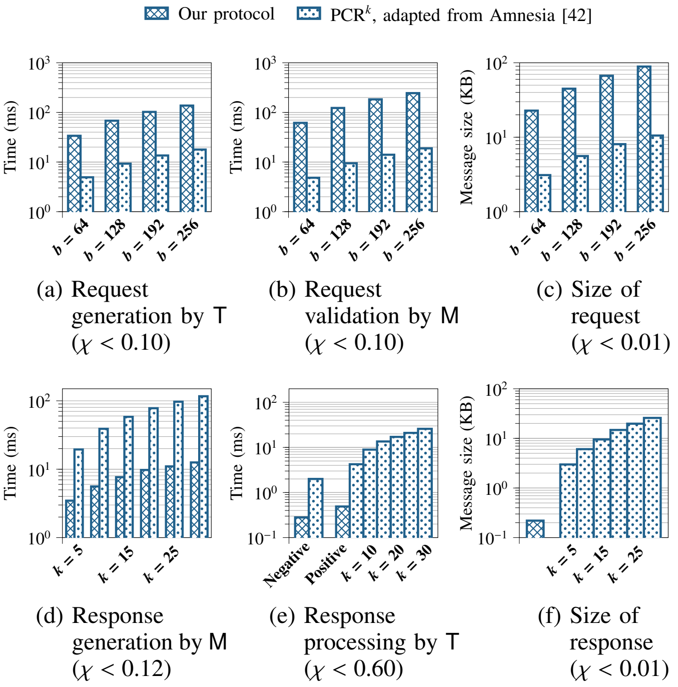
A fairly direct method to adapt the PCR protocol of Amnesia to accommodate the local-detection design in Sec. V-A using Bloom filter $\langle \mathcal{F}, \mathcal{B} \rangle$ is for T to include the elements of $\mathcal{B}$ in the encrypted data structure it sends to M, instead of the (hashes of the) account passwords themselves. Upon receiving a login attempt with password π, M can then execute the response computation k times, once using $f_i(H(\pi))$ as the test plaintext for $i = 1 \ldots k$. As long as the response plaintexts in these executions were chosen to combine to produce π (say, by a k-out-of-k secret sharing), T would obtain π if and only if $H(\pi) \in_{\mathrm{B}} \langle \mathcal{F}, \mathcal{B} \rangle$. We refer to this adaptation using k executions of the PCR protocol as PCR^k.

We implemented PCR^k as a simple modification to the Go implementation of the original Amnesia remote-monitoring protocol. The comparison between our protocol of Sec. V-B and PCR^k is given in Fig. 8; note that all vertical axes are log-scale. As before, all datapoints are the means of 50 runs; we report relative standard deviations (χ) in each caption.

Monitor deployment is a rare cost compared to monitor responses, as one monitor deployment could produce thousands of monitor responses during its lifetime at M. So, while the monitor request is larger in our design than in PCR^k (Fig. 8c) and the monitor request is an order-of-magnitude more costly for T to create (Fig. 8a) and M to validate (Fig. 8b), these costs are still modest ($\leq$ 200ms in all cases shown) and concern us little. Moreover, $\approx$ 79% of the monitor request size is consumed by the proof Ψ, which is not saved once it is checked (line t5); as such, the storage consumed by the saved monitor request at M is similar to that in PCR^k. The far more important costs are response generation by M (Fig. 8d), response processing by T (Fig. 8e), and the response message size (Fig. 8f), since these are incurred per unsuccessful login at M. As we can see, our protocol outperforms PCR^k by an order of magnitude in all of these measures.

Appendix B
Security Analysis of Remote Monitoring Protocol

In this appendix, we prove security for the protocol described in Sec. V-B2. We start by defining the primitives on which security relies, in Sec. B-A. We then prove security against a malicious target site T in Sec. B-B, and against a malicious monitor site M in Sec. B-C.



(a) Request generation by T ($\chi < 0.10$)

(b) Request validation by M ($\chi < 0.10$)

(c) Size of request ($\chi < 0.01$)

(d) Response generation by M ($\chi < 0.12$)

(e) Response processing by T ($\chi < 0.60$)

(f) Size of response ($\chi < 0.01$)

Fig. 8: Performance comparison between our protocol (Sec. V-B2) and PCR^k. One-time costs for T to deploy a monitoring request to M are shown in (a)–(c), and costs for M to return a response to T, once per incorrect login attempt, are shown in (d)–(f). In (e), "negative" refers to the case $H(\pi) \notin_{\mathrm{B}} \langle \mathcal{F}, \mathcal{B} \rangle$, and "positive" refers to the case $H(\pi) \in_{\mathrm{B}} \langle \mathcal{F}, \mathcal{B} \rangle$. In PCR^k, the "positive" cost depends on k. Request generation, validation, and size in PCR^k also depends on $b' = |\mathcal{B}|$; numbers shown are for $b' = b/2$ in (a)–(c).

A. Primitives

In this section we provide definitions for the primitives used in our cryptographic protocol described in Sec. V-B.

1) IND-CPA secure encryption: Our protocol relies on a partially homomorphic encryption scheme $\mathcal{E}$ achieving indistinguishability under chosen plaintext attack (IND-CPA) security [3]. We define the IND-CPA experiment $\mathsf{Expt}_{\mathcal{E}}^{\text{CPA},\mathsf{b}'}$ as:

```
Experiment $\mathsf{Expt}_{\mathcal{E}}^{\text{CPA},\mathsf{b}'}(\mathfrak{D}_{\text{CPA}})$
    $\langle pk, sk \rangle \leftarrow \mathsf{Gen}()$
    $\hat{\mathsf{b}'} \leftarrow \mathfrak{D}_{\text{CPA}}^{\mathsf{Enc}_{pk}(\mathrm{LR}(\cdot,\cdot,\mathsf{b}'))}(pk)$
    return $\hat{\mathsf{b}'}$
```

The IND-CPA adversary $\mathfrak{D}_{\text{CPA}}$ is given access to a "left-or-right" oracle $\mathsf{Enc}_{pk}(\mathrm{LR}(\cdot,\cdot,\mathsf{b}'))$ that takes two plaintexts m_0, m_1 as inputs and returns $\mathsf{Enc}_{pk}(m_{\mathsf{b}'})$. Finally, $\mathfrak{D}_{\text{CPA}}$ returns a bit $\hat{\mathsf{b}'}$, which the experiment returns. We define

$$\mathsf{Adv}_{\mathcal{E}}^{\text{CPA}}(\mathfrak{D}_{\text{CPA}}) = \mathbb{P}\left(\mathsf{Expt}_{\mathcal{E}}^{\text{CPA},0}(\mathfrak{D}_{\text{CPA}}) = 0\right) - \mathbb{P}\left(\mathsf{Expt}_{\mathcal{E}}^{\text{CPA},1}(\mathfrak{D}_{\text{CPA}}) = 0\right)$$
$$\mathsf{Adv}_{\mathcal{E}}^{\text{CPA}}(t_{\text{CPA}}, q_{\text{CPA}}) = \max_{\mathfrak{D}_{\text{CPA}}} \mathsf{Adv}_{\mathcal{E}}^{\text{CPA}}(\mathfrak{D}_{\text{CPA}})$$

where the maximum is taken over all IND-CPA adversaries $\mathfrak{D}_{\text{CPA}}$ running in time t_{CPA} and making up to q_{CPA} oracle queries.

ElGamal encryption [16] can be used to instantiate the homomorphic encryption scheme $\mathcal{E}$ in Sec. V-B2. Given are a cyclic group G of prime order q and a generator g of G.

- $\mathsf{Gen}()$ returns a key pair $\langle pk, sk \rangle$, including a private key $sk = \langle u \rangle$ and a public key $pk = \langle G, g, U \rangle$, where $u \xleftarrow{\$} \mathbb{Z}_q$ and $U \leftarrow g^u$.
- $\mathsf{Enc}_{\langle G,g,U \rangle}(m)$ returns $\langle V, W \rangle$ where $V \leftarrow g^v$, $v \xleftarrow{\$} \mathbb{Z}_q$, and $W \leftarrow mU^v$ for a plaintext $m \in G$.
- $\mathsf{Dec}_{\langle u \rangle}(\langle V, W \rangle)$ returns $m \leftarrow WV^{-u}$.
- $\langle V_1, W_1 \rangle \times_{\langle G,g,U \rangle} \langle V_2, W_2 \rangle$ returns $\langle V_1 V_2 g^y, W_1 W_2 U^y \rangle$ for $y \xleftarrow{\$} \mathbb{Z}_q$ if $\{V_1, W_1, V_2, W_2\} \subseteq G$ and returns $\perp$ otherwise.

The IND-CPA security of ElGamal encryption was proved by Tsiounis and Yung [36]. Our protocols descriptions leave implicit the checks needed to determine whether ciphertexts are well-formed, but Prop. 1 indicates that these are trivial.

Proposition 1. *For ElGamal encryption, $C_{\langle G,g,U \rangle} = G \times G$.*

Proof: $C_{\langle G,g,U \rangle} \subseteq G \times G$ follows from the fact that G is a cyclic group. Also, given that for any $\langle V, W \rangle \in G \times G$, there exists $m \in G$ such that $m = WV^{-u}$ for $u \in \mathbb{Z}_q$. Therefore $\langle V, W \rangle \in C_{\langle G,g,U \rangle}(m)$. ■

In addition, ElGamal is well-known to be multiplicatively homomorphic, which is confirmed in Prop. 2.

Proposition 2. *For ElGamal encryption, if $c_1 \in C_{\langle G,g,U \rangle}(m_1)$ and $c_2 \in C_{\langle G,g,U \rangle}(m_2)$, then $c_1 \times_{\langle G,g,U \rangle} c_2$ is uniformly distributed in $C_{\langle G,g,U \rangle}(m_1 m_2)$.*

Proof: Let $c_1 = \langle V_1, W_1 \rangle$ and $c_2 = \langle V_2, W_2 \rangle$. For $V = V_1 V_2 = g^{v_1+v_2}$, $W = m_1 m_2 U^{v_1+v_2}$, and for any $y \xleftarrow{\$} \mathbb{Z}_q$,

$$\langle V_1, W_1 \rangle \times_{\langle G,g,U \rangle} \langle V_2, W_2 \rangle = \langle Vg^y, WU^y \rangle$$

which is a re-randomization of $\langle V, W \rangle$ and is uniformly distributed in $C_{\langle G,g,U \rangle}(m_1 m_2)$. ■

2) Noninteractive zero-knowledge proofs: Our protocol in Sec. V-B additionally leverages a noninteractive zero-knowledge proof of membership for an NP language $\mathcal{L}$, implemented by scheme $\Pi = (\mathsf{zkpGen}, \mathsf{zkpVerify}, \mathsf{zkpSim})$, in the random oracle model [4]. zkpSim offers two interfaces, denoted $\mathsf{zkpSim.hash}$ and $\mathsf{zkpSim.prove}$, that share state between them. Let $\mathcal{R}_{\mathcal{L}}$ be the witness relation for $\mathcal{L}$, and let $\mathcal{H}$ denote the set of all functions from $\{0,1\}^*$ to $\{0,1\}^\infty$. On input $(\mathrm{x}, w) \in \mathcal{R}_{\mathcal{L}}$ and with access to a random oracle $\mathsf{hash} \xleftarrow{\$} \mathcal{H}$, $\mathsf{zkpGen}_w^{\mathsf{hash}}(\mathrm{x})$ produces a proof Ψ (using the witness w) that $\mathrm{x} \in \mathcal{L}$, so that if $\Psi \leftarrow \mathsf{zkpGen}_w^{\mathsf{hash}}(\mathrm{x})$ then $\mathsf{zkpVerify}^{\mathsf{hash}}(\mathrm{x}, \Psi)$ returns true. We reduce the security of our protocol to the following adversary advantages against Π.

a) Soundness advantage: For any $\widehat{\mathsf{zkpGen}}$, the soundness advantage is

$$\mathsf{Adv}_{\Pi}^{\text{SND}}(\widehat{\mathsf{zkpGen}}) = \max_{\mathrm{x} \notin \mathcal{L}} \mathbb{P}\left(\mathsf{zkpVerify}^{\mathsf{hash}}(\mathrm{x}, \widehat{\mathsf{zkpGen}}_{\perp}^{\mathsf{hash}}(\mathrm{x}))\right)$$

For any time t_{SND}, the soundness advantage is

$$\mathsf{Adv}_{\Pi}^{\text{SND}}(t_{\text{SND}}) = \max_{\widehat{\mathsf{zkpGen}}} \mathsf{Adv}_{\Pi}^{\text{SND}}(\widehat{\mathsf{zkpGen}})$$

where the maximum is taken over all algorithms $\widehat{\mathsf{zkpGen}}$ running in time t_{SND}.

b) Distinguishing advantage: We define a distinguishing adversary to be an algorithm $\mathfrak{D}_{\text{ZKP}}$ that can participate in either of the experiments $\mathsf{Expt}_{\Pi}^{\text{ZKP},\mathsf{b}''}$ defined below:

```
Experiment $\mathsf{Expt}_{\Pi}^{\text{ZKP},0}(\mathfrak{D}_{\text{ZKP}})$
    $(\mathrm{x}, w) \xleftarrow{\$} \mathcal{R}_{\mathcal{L}}$
    $\mathsf{hash} \xleftarrow{\$} \mathcal{H}$
    $\Psi \leftarrow \mathsf{zkpGen}_{w}^{\mathsf{hash}}(\mathrm{x})$
    $\hat{\mathsf{b}''} \leftarrow \mathfrak{D}_{\text{ZKP}}^{\mathsf{hash}}(\mathrm{x}, \Psi)$
    return $\hat{\mathsf{b}''}$

Experiment $\mathsf{Expt}_{\Pi}^{\text{ZKP},1}(\mathfrak{D}_{\text{ZKP}})$
    $(\mathrm{x}, w) \xleftarrow{\$} \mathcal{R}_{\mathcal{L}}$
    $\Psi \leftarrow \mathsf{zkpSim.prove}(\mathrm{x})$
    $\hat{\mathsf{b}''} \leftarrow \mathfrak{D}_{\text{ZKP}}^{\mathsf{zkpSim.hash}}(\mathrm{x}, \Psi)$
    return $\hat{\mathsf{b}''}$
```

In words, the adversary $\mathfrak{D}_{\text{ZKP}}$ must distinguish between a real proof output from $\mathsf{zkpGen}_{w}^{\mathsf{hash}}(\mathrm{x})$ and a proof output from the simulator $\mathsf{zkpSim.prove}(\mathrm{x})$ without knowledge of the witness w but with the ability to implement the hash function $\mathsf{zkpSim.hash}$. This permits zkpSim to leverage the standard technique of "backpatching" the random oracle outputs on inputs that $\mathfrak{D}_{\text{ZKP}}$ has not yet queried. We define

$$\mathsf{Adv}_{\Pi}^{\text{ZKP}}(\mathfrak{D}_{\text{ZKP}}) = \mathbb{P}\left(\mathsf{Expt}_{\Pi}^{\text{ZKP},1}(\mathfrak{D}_{\text{ZKP}}) = 1\right) - \mathbb{P}\left(\mathsf{Expt}_{\Pi}^{\text{ZKP},0}(\mathfrak{D}_{\text{ZKP}}) = 1\right)$$
$$\mathsf{Adv}_{\Pi}^{\text{ZKP}}(t_{\text{ZKP}}, q_{\text{RO}}) = \max_{\mathfrak{D}_{\text{ZKP}}} \mathsf{Adv}_{\Pi}^{\text{ZKP}}(\mathfrak{D}_{\text{ZKP}})$$

where the maximum is taken over all adversaries $\mathfrak{D}_{\text{ZKP}}$ making q_{RO} random oracle queries and running in time t_{ZKP}.

Our implementation leverages a zero-knowledge proof of the equality of discrete logarithms, due to Chaum and Pedersen [10]. More specifically, this technique demonstrates that an ElGamal ciphertext $\langle V, W\rangle$ satisfies $\langle V, W\rangle \in C_{\langle G,g,U\rangle}(g)$ by proving $\log_g(V) = \log_U(Wg^{-1})$ in zero knowledge, and similarly for a ciphertext $\langle V, W\rangle$ satisfying $\langle V, W\rangle \in C_{\langle G,g,U\rangle}(g^{-1})$. We combine these zero-knowledge proof techniques to demonstrate only that $\langle V, W\rangle \in C_{\langle G,g,U\rangle}(g) \cup C_{\langle G,g,U\rangle}(g^{-1})$ in zero knowledge using a technique due to Cramer et al. [11].

B. Security against a Malicious T

In this section we prove that a malicious T learns nothing about π from the response computed by an honest M unless T already guessed $H(\pi)$, in the sense that $H(\pi) \in_{\text{B}} \langle \mathcal{F}, \mathcal{B}'\rangle$ for the Bloom filter $\langle \mathcal{F}, \mathcal{B}'\rangle$ encoded in its request. An important premise here is that pk is a valid public key of the underlying cryptosystem, which is implicitly assumed to be verified by M upon receiving message r1, and that the ciphertexts received in message r1 are valid for the cryptosystem; for the cryptosystem used in our implementation, this can be easily verified (see Prop. 1). Then, the following propositions show that a malicious T learns nothing about π if any of the following three conditions is not satisfied: $\{c_j\}_{j=1}^{b} \subseteq C_{pk}(g) \cup C_{pk}(g^{-1})$, $|\mathcal{B}'| = b'$, or $H(\pi) \in_{\text{B}} \langle \mathcal{F}, \mathcal{B}'\rangle$.

Proposition 3. *If* $\langle pk, \mathcal{F}, b', \{c_j\}_{j=1}^{b}, \Psi\rangle$*, where* $\{c_j\}_{j=1}^{b} \not\subseteq \left(C_{pk}(g) \cup C_{pk}(g^{-1})\right)$*, is produced by a* T*-adversary* $\mathfrak{T}$ *running in time* t_{SND}*, then* M *fails to abort in line* m1 *with probability at most* $\mathsf{Adv}_{\Pi}^{\text{SND}}(t_{\text{SND}})$*.*

Proof: This is immediate from the definition of soundness advantage. ■

Proposition 4. *Suppose* M *receives* $\langle pk, \mathcal{F}, b', \{c_j\}_{j=1}^{b}, \Psi\rangle$ *where* $\{c_j\}_{j=1}^{b} \subseteq C_{pk}(g) \cup C_{pk}(g^{-1})$*, and let* $\mathcal{B}' = \{j : c_j \in C_{pk}(g)\}$*. If* $|\mathcal{B}'| \neq b'$*, then for any* $m, m' \in \mathcal{G}$*,*

$$\mathbb{P}\left(\begin{array}{r} \hat{c}_0 \in C_{pk}(m) \\ \wedge\ \hat{c}_1 \in C_{pk}(m') \end{array} \middle| \begin{array}{r} \{c_j\}_{j=1}^{b} \subseteq C_{pk}(g) \cup C_{pk}(g^{-1}) \\ \wedge\ |\mathcal{B}'| \neq b' \end{array}\right) \le \frac{1}{q-1}$$

Proof: First note that $\hat{c}_0 \in C_{pk}(\mathbf{1}) \Leftrightarrow \hat{c}_1 \in C_{pk}(\pi)$. So, it suffices to quantify the probability in the proposition for the cases $m = \mathbf{1} \wedge m' = \pi$ and $m \neq \mathbf{1} \wedge m' \neq \pi$. If $\{c_j\}_{j=1}^{b} \subseteq C_{pk}(g) \cup C_{pk}(g^{-1})$ but $|\mathcal{B}'| \neq b'$, then $d_0 \in C_{pk} \setminus C_{pk}(\mathbf{1})$ in line m4. We consider two cases.

- First suppose $d_1 \in C_{pk} \setminus C_{pk}(\mathbf{1})$. For the case $m = \mathbf{1}$ and $m' = \pi$,

$$\begin{aligned} &\mathbb{P}\left(\begin{array}{r} \hat{c}_0 \in C_{pk}(\mathbf{1}) \\ \wedge\ \hat{c}_1 \in C_{pk}(\pi) \end{array} \middle| \begin{array}{r} d_0 \in C_{pk} \setminus C_{pk}(\mathbf{1}) \\ \wedge\ d_1 \in C_{pk} \setminus C_{pk}(\mathbf{1}) \end{array}\right) \\ &= \sum_{\hat{m} \in \mathcal{G} \setminus \{\mathbf{1}\}} \mathbb{P}\left(\begin{array}{r} \$_{pk}(d_0) \in C_{pk}(\hat{m}) \\ \wedge\ \$_{pk}(d_1) \in C_{pk}(\hat{m}^{-1}) \end{array} \middle| \begin{array}{r} d_0 \in C_{pk} \setminus C_{pk}(\mathbf{1}) \\ \wedge\ d_1 \in C_{pk} \setminus C_{pk}(\mathbf{1}) \end{array}\right) \\ &= (q-1)\frac{1}{q-1}\frac{1}{q-1} = \frac{1}{q-1} \end{aligned}$$

  And for any $m \neq \mathbf{1}$ and $m' \neq \pi$,

$$\begin{aligned} &\mathbb{P}\left(\begin{array}{r} \hat{c}_0 \in C_{pk}(m) \\ \wedge\ \hat{c}_1 \in C_{pk}(m') \end{array} \middle| \begin{array}{r} d_0 \in C_{pk} \setminus C_{pk}(\mathbf{1}) \\ \wedge\ d_1 \in C_{pk} \setminus C_{pk}(\mathbf{1}) \end{array}\right) \\ &= \sum_{\hat{m} \in \mathcal{G} \setminus \{\mathbf{1}, m\}} \mathbb{P}\left(\begin{array}{r} \$_{pk}(d_0) \in C_{pk}(\hat{m}) \\ \wedge\ \$_{pk}(d_1) \in C_{pk}(m \circ \hat{m}^{-1}) \\ \wedge\ \$_{pk}(\hat{c}_0) \in C_{pk}(m' \circ \pi^{-1}) \end{array} \middle| \begin{array}{r} d_0 \in C_{pk} \setminus C_{pk}(\mathbf{1}) \\ \wedge\ d_1 \in C_{pk} \setminus C_{pk}(\mathbf{1}) \end{array}\right) \\ &= (q-2)\frac{1}{q-1}\frac{1}{q-1}\frac{1}{q-1} = \frac{q-2}{(q-1)^3} \end{aligned}$$

- Now suppose $d_1 \in C_{pk}(\mathbf{1})$. In this case $m = \mathbf{1}$ is not possible, since $\$_{pk}(d_0) \times_{pk} \$_{pk}(d_1) \in C_{pk} \setminus C_{pk}(\mathbf{1})$. For any $m \neq \mathbf{1}$ and $m' \neq \pi$,

$$\begin{aligned} &\mathbb{P}\left(\begin{array}{r} \hat{c}_0 \in C_{pk}(m) \\ \wedge\ \hat{c}_1 \in C_{pk}(m') \end{array} \middle| \begin{array}{r} d_0 \in C_{pk} \setminus C_{pk}(\mathbf{1}) \\ \wedge\ d_1 \in C_{pk}(\mathbf{1}) \end{array}\right) \\ &= \mathbb{P}\left(\begin{array}{r} \$_{pk}(d_0) \in C_{pk}(m) \\ \wedge\ \$_{pk}(\hat{c}_0) \in C_{pk}(m' \circ \pi^{-1}) \end{array} \middle| \begin{array}{r} d_0 \in C_{pk} \setminus C_{pk}(\mathbf{1}) \\ \wedge\ d_1 \in C_{pk}(\mathbf{1}) \end{array}\right) \\ &= \frac{1}{q-1}\frac{1}{q-1} = \frac{1}{(q-1)^2} \end{aligned}$$

■

Proposition 5. *Suppose* M *receives* $\langle pk, \mathcal{F}, b', \{c_j\}_{j=1}^{b}, \Psi\rangle$ *where* $\{c_j\}_{j=1}^{b} \subseteq C_{pk}(g) \cup C_{pk}(g^{-1})$*, and let* $\mathcal{B}' = \{j : c_j \in C_{pk}(g)\}$*. If* $|\mathcal{B}'| = b'$ *but* $H(\pi) \notin_{\text{B}} \langle \mathcal{F}, \mathcal{B}'\rangle$*, then for any* $m, m' \in \mathcal{G}$*,*

$$\mathbb{P}\left(\begin{array}{r} \hat{c}_0 \in C_{pk}(m) \\ \wedge\ \hat{c}_1 \in C_{pk}(m') \end{array} \middle| \begin{array}{r} \{c_j\}_{j=1}^{b} \subseteq C_{pk}(g) \cup C_{pk}(g^{-1}) \\ \wedge\ |\mathcal{B}'| = b' \wedge H(\pi) \notin_{\text{B}} \langle \mathcal{F}, \mathcal{B}'\rangle \end{array}\right) \le \frac{1}{(q-1)^2}$$

Proof: First note that $\hat{c}_0 \in C_{pk}(\mathbf{1}) \Leftrightarrow \hat{c}_1 \in C_{pk}(\pi)$. So, it suffices to quantify the probability in the proposition for the cases $m = \mathbf{1} \wedge m' = \pi$ and $m \neq \mathbf{1} \wedge m' \neq \pi$. If $\{c_j\}_{j=1}^{b} \subseteq C_{pk}(g) \cup C_{pk}(g^{-1})$ and $|\mathcal{B}'| = b'$ but $H(\pi) \notin_{\text{B}} \langle \mathcal{F}, \mathcal{B}'\rangle$, then $d_0 \in C_{pk}(\mathbf{1})$ but in line m6, $d_1 \in C_{pk}(g^{-2|\mathcal{F}(H(\pi)) \setminus \mathcal{B}'|})$ and so $d_1 \in C_{pk} \setminus C_{pk}(\mathbf{1})$. In this case $m = \mathbf{1}$ is not possible, since

$\$_{pk}(d_0) \times_{pk} \$_{pk}(d_1) \in C_{pk} \setminus C_{pk}(\mathbf{1})$. For any $m \neq \mathbf{1}$ and $m' \neq \pi$,

$$\begin{aligned}
&\mathbb{P}\left(\begin{array}{r} \hat{c}_0 \in C_{pk}(m) \\ \wedge\ \hat{c}_1 \in C_{pk}(m') \end{array} \middle| \begin{array}{r} d_0 \in C_{pk}(\mathbf{1}) \\ \wedge\ d_1 \in C_{pk} \setminus C_{pk}(\mathbf{1}) \end{array}\right) \\
&= \mathbb{P}\left(\begin{array}{r} \$_{pk}(d_1) \in C_{pk}(m) \\ \wedge\ \$_{pk}(\hat{c}_0) \in C_{pk}(m' \circ \pi^{-1}) \end{array} \middle| \begin{array}{r} d_0 \in C_{pk}(\mathbf{1}) \\ \wedge\ d_1 \in C_{pk} \setminus C_{pk}(\mathbf{1}) \end{array}\right) \\
&= \frac{1}{q-1}\frac{1}{q-1} = \frac{1}{(q-1)^2}
\end{aligned}$$

■

A malicious T without knowledge of $H(\pi)$ can increase the probability of $H(\pi) \in_{\mathrm{B}} \langle \mathcal{F}, \mathcal{B}' \rangle$ by increasing $|\mathcal{B}'|$ (or b', as the malicious T must ensure $|\mathcal{B}'| = b'$, by Prop. 4). In practice, M could determine an acceptable threshold and drop monitoring requests for which b' exceeds that threshold.

C. Security against a malicious M

We need to show that message r1 does not leak information about T's input $\mathcal{B}$ (except its size b'), assuming T is honest. More precisely, we consider the following experiment to characterize success of a malicious M in distinguishing between two Bloom filters $\langle \mathcal{F}, \mathcal{B}_0 \rangle$ and $\langle \mathcal{F}, \mathcal{B}_1 \rangle$ for $\mathcal{B}_0$, $\mathcal{B}_1$ of its own choosing (but of the same size, and each containing $\mathcal{F}(\hat{\pi})$ for the user-chosen password $\hat{\pi}$), based on message r1.

```
Experiment $\mathsf{Expt}^{\mathsf{T},\mathsf{b}}(\langle \mathfrak{M}_1, \mathfrak{M}_2 \rangle)$
    $\mathsf{hash} \xleftarrow{\$} \mathcal{H}$
    $\langle \mathcal{B}_0, \mathcal{B}_1, \phi \rangle \leftarrow \mathfrak{M}_1^{\mathsf{hash}}(\mathcal{F})$
    $\langle pk, \mathcal{F}, b', \{c_j\}_{j=1}^{b}, \Psi \rangle \leftarrow \mathsf{T}_{\mathsf{t1\text{-}t4}}(\langle \mathcal{F}, \mathcal{B}_{\mathsf{b}} \rangle)$
    $\hat{\mathsf{b}} \leftarrow \mathfrak{M}_2^{\mathsf{hash}}(\langle pk, \mathcal{F}, b', \{c_j\}_{j=1}^{b}, \Psi \rangle, \phi)$
    return $\hat{\mathsf{b}}$
```

Here, $\mathsf{T}_{\mathsf{t1\text{-}t4}}$ denotes lines t1–t4 in Fig. 5, and $\mathcal{F}$ is assumed to be sampled according the Bloom filter algorithm's specification. We define the M-adversary advantage as

$$\mathsf{Adv}^{\mathsf{T}}(\mathfrak{M}) = \mathbb{P}\left(\mathsf{Expt}^{\mathsf{T},0}(\mathfrak{M}) = 0\right) - \mathbb{P}\left(\mathsf{Expt}^{\mathsf{T},1}(\mathfrak{M}) = 0\right)$$

$$\mathsf{Adv}^{\mathsf{T}}(t, q_{\mathrm{RO}}) = \max_{\mathfrak{M}} \mathsf{Adv}^{\mathsf{T}}(\mathfrak{M})$$

where the maximum is taken over all adversaries $\mathfrak{M}$ that make at most q_{RO} random oracle queries and execute in time t. In proving time bounds on adversaries, we ignore constant terms.

Proposition 6.

$$\mathsf{Adv}^{\mathsf{T}}(t, q_{\mathrm{RO}}) \leq b \cdot \mathsf{Adv}_{\mathcal{E}}^{\mathrm{CPA}}(t_{\mathrm{CPA}}, q_{\mathrm{CPA}}) + 2 \cdot \mathsf{Adv}_{\Pi}^{\mathrm{ZKP}}(t_{\mathrm{ZKP}}, q_{\mathrm{ZKP}}) \ ,$$

where $q_{\mathrm{CPA}} = 1$*,* $q_{\mathrm{ZKP}} \leq q_{\mathrm{RO}}$*,* $t_{\mathrm{CPA}} \leq t + b \cdot t_{\mathrm{ENC}} + q_{\mathrm{RO}} \cdot t_{\mathrm{HASH}}$*,* $t_{\mathrm{ZKP}} \leq t + b \cdot t_{\mathrm{ENC}} + q_{\mathrm{RO}} \cdot t_{\mathrm{HASH}}$*, and* t_{ENC} *and* t_{HASH} *are the times for one invocation of* Enc *and* hash*, respectively.*

Proof: Since M observes ciphertexts $\{c_j\}_{j=1}^{b}$ and a noninteractive zero-knowledge proof Ψ, and other components including pk and $\mathcal{F}$ that do not depend on $\mathcal{B}$, we reduce the advantage of a M-adversary $\mathfrak{M}$ to the IND-CPA advantage of the encryption scheme $\mathcal{E}$ and the distinguishing advantage of the noninteractive zero-knowledge proof Ψ.

To do this, let $\mathfrak{M} = \langle \mathfrak{M}_1, \mathfrak{M}_2 \rangle$ be a M-adversary and define a sequence of experiments for $\mathfrak{M}$

$$\mathsf{Expt}^{\mathsf{T},00}(\mathfrak{M}), \quad \mathsf{Expt}^{\mathsf{T},01}(\mathfrak{M}), \quad \mathsf{Expt}^{\mathsf{T},11}(\mathfrak{M}), \quad \mathsf{Expt}^{\mathsf{T},10}(\mathfrak{M})$$

where we associate each pair α,β of bits to a (α,β) hybrid experiment as shown in Fig. 9. In words, hybrid experiment $\mathsf{Expt}^{\mathsf{T},\alpha\beta}(\mathfrak{M})$ produces ciphertexts $\{c_j\}_{j=1}^{b}$ based on $\langle \mathcal{F}, \mathcal{B}_\alpha \rangle$ and, depending on whether $\beta = 0$ or $\beta = 1$, generates a real[4] Ψ or a simulated Ψ for the statement that $\{c_j\}_{j=1}^{b} \subseteq C_{pk}(g) \cup C_{pk}(g^{-1})$. So if we let $\mathcal{P}(\alpha,\beta) = \mathbb{P}\left(\mathsf{Expt}^{\mathsf{T},\alpha\beta}(\mathfrak{M}) = 0\right)$ for bits $\alpha,\beta \in \{0, 1\}$, then it will be the case that

$$\begin{aligned}
\mathcal{P}(0,0) &= \mathbb{P}\left(\mathsf{Expt}^{\mathsf{T},00}(\mathfrak{M}) = 0\right) = \mathbb{P}\left(\mathsf{Expt}^{\mathsf{T},0}(\mathfrak{M}) = 0\right) \\
\mathcal{P}(1,0) &= \mathbb{P}\left(\mathsf{Expt}^{\mathsf{T},10}(\mathfrak{M}) = 0\right) = \mathbb{P}\left(\mathsf{Expt}^{\mathsf{T},1}(\mathfrak{M}) = 0\right)
\end{aligned}$$

and so we have:

$$\begin{aligned}
&\mathsf{Adv}^{\mathsf{T}}(\mathfrak{M}) \\
&= \mathbb{P}\left(\mathsf{Expt}^{\mathsf{T},0}(\mathfrak{M}) = 0\right) - \mathbb{P}\left(\mathsf{Expt}^{\mathsf{T},1}(\mathfrak{M}) = 0\right) \\
&= \mathcal{P}(0,0) - \mathcal{P}(1,0) \\
&= \mathcal{P}(0,0) - \mathcal{P}(0,1) + \mathcal{P}(0,1) - \mathcal{P}(1,1) + \mathcal{P}(1,1) - \mathcal{P}(1,0) \quad (24)
\end{aligned}$$

We now show that:

$$\begin{aligned}
\mathcal{P}(0,1) - \mathcal{P}(1,1) &\leq b \cdot \mathsf{Adv}_{\mathcal{E}}^{\mathrm{CPA}}(t_{\mathrm{CPA}}, q_{\mathrm{CPA}}) && (25) \\
\mathcal{P}(0,0) - \mathcal{P}(0,1) &\leq \mathsf{Adv}_{\Pi}^{\mathrm{ZKP}}(t_{\mathrm{ZKP}}, q_{\mathrm{ZKP}}) && (26) \\
\mathcal{P}(1,1) - \mathcal{P}(1,0) &\leq \mathsf{Adv}_{\Pi}^{\mathrm{ZKP}}(t_{\mathrm{ZKP}}, q_{\mathrm{ZKP}}) \ , && (27)
\end{aligned}$$

where, for all adversaries $\mathfrak{M}$ that make at most q_{RO} random oracle queries and execute in total time t, we have $q_{\mathrm{CPA}} = 1$, $q_{\mathrm{ZKP}} \leq q_{\mathrm{RO}}$, $t_{\mathrm{CPA}} \leq t + b \cdot t_{\mathrm{ENC}} + q_{\mathrm{RO}} \cdot t_{\mathrm{HASH}}$, and $t_{\mathrm{ZKP}} \leq t + b \cdot t_{\mathrm{ENC}} + q_{\mathrm{RO}} \cdot t_{\mathrm{HASH}}$. So, combining (25), (26), and (27) with (24), we have

$$\mathsf{Adv}^{\mathsf{T}}(t, q_{\mathrm{RO}}) \leq b \cdot \mathsf{Adv}_{\mathcal{E}}^{\mathrm{CPA}}(t_{\mathrm{CPA}}, q_{\mathrm{CPA}}) + 2 \cdot \mathsf{Adv}_{\Pi}^{\mathrm{ZKP}}(t_{\mathrm{ZKP}}, q_{\mathrm{ZKP}}) \ .$$

Justification of (25): Given M-adversary $\mathfrak{M} = \langle \mathfrak{M}_1, \mathfrak{M}_2 \rangle$, we construct an IND-CPA adversary $\mathfrak{D}_{\mathrm{CPA}}$ attacking the IND-CPA experiment $\mathsf{Expt}_{\mathcal{E}}^{\mathrm{CPA},\mathsf{b}'}$ defined in Sec. B-A1. $\mathfrak{D}_{\mathrm{CPA}}$ first invokes $\mathfrak{M}_1$ with $\mathcal{F}$, servicing its oracle queries using zkpSim.hash, and receives $\mathcal{B}_0$ and $\mathcal{B}_1$ (aborting if they are of unequal size) from $\mathfrak{M}_1$. $\mathfrak{D}_{\mathrm{CPA}}$ then sets $m_{0j} \leftarrow g$ if $j \in \mathcal{B}_0$ and $m_{0j} \leftarrow g^{-1}$ otherwise, and $m_{1j} \leftarrow g$ if $j \in \mathcal{B}_1$ and $m_{1j} \leftarrow g^{-1}$ otherwise. $\mathfrak{D}_{\mathrm{CPA}}$ chooses an index uniformly at random $i \xleftarrow{\$} \{1, ..., b\}$ and computes $\{c_j\}_{j=1}^{b}$ as follows:

- For $j < i$, $\mathfrak{D}_{\mathrm{CPA}}$ computes $c_j \leftarrow \mathsf{Enc}_{pk}(m_{0j})$
- For $j = i$, $\mathfrak{D}_{\mathrm{CPA}}$ queries its oracle and obtains $c_j \leftarrow \mathsf{Enc}_{pk}(\mathrm{LR}(m_{0j}, m_{1j}, \mathsf{b}'))$
- For $j > i$, $\mathfrak{D}_{\mathrm{CPA}}$ computes $c_j \leftarrow \mathsf{Enc}_{pk}(m_{1j})$

$\mathfrak{D}_{\mathrm{CPA}}$ also executes $\mathsf{zkpSim.prove}(\langle pk, \{c_j\}_{j=1}^{b} \rangle)$ and generates a simulated noninteractive zero-knowledge proof Ψ for $\{c_j\}_{j=1}^{b} \subseteq C_{pk}(g) \cup C_{pk}(g^{-1})$. Then $\mathfrak{D}_{\mathrm{CPA}}$ invokes $\mathfrak{M}_2$ with $\langle pk, \mathcal{F}, b', \{c_j\}_{j=1}^{b}, \Psi \rangle$ and services its oracle queries using zkpSim.hash. Finally $\mathfrak{D}_{\mathrm{CPA}}$ returns the bit $\hat{\mathsf{b}}$ returned by $\mathfrak{M}_2$ as its guess $\hat{\mathsf{b}'}$. Here we use $i \in \{1, ..., b\}$ to index the experiment simulated by $\mathfrak{D}_{\mathrm{CPA}}$ for $\mathsf{Expt}_i^{\mathsf{T},\mathsf{b}'1}(\mathfrak{M})$ and we let $\mathsf{H}(\mathsf{b}', i) = \mathsf{Expt}_i^{\mathsf{T},\mathsf{b}'1}(\mathfrak{M})$. Then, $\mathsf{H}(0, b) \overset{d}{=} \mathsf{Expt}^{\mathsf{T},01}(\mathfrak{M})$, $\mathsf{H}(1, 1) \overset{d}{=} \mathsf{Expt}^{\mathsf{T},11}(\mathfrak{M})$ and, for $2 \leq i \leq b$, $\mathsf{H}(0, i-1) \overset{d}{=} \mathsf{H}(1, i)$. Given some fixed choice of $i = i^*$, the simulation provided by $\mathfrak{D}_{\mathrm{CPA}}$ to $\mathfrak{M}$ is perfectly

[4] zkpGen leverages the witness produced by $\mathsf{T}_{\mathsf{t1\text{-}t3}}$ to do so.

```
Experiment $\mathsf{Expt}^{\mathsf{T},00}(\langle\mathfrak{M}_1,\mathfrak{M}_2\rangle)$
    $\mathsf{hash} \xleftarrow{\$} \mathcal{H}$
    $\langle\mathcal{B}_0,\mathcal{B}_1,\phi\rangle \leftarrow \mathfrak{M}_1^{\mathsf{hash}}(\mathcal{F})$
    if $|\mathcal{B}_0| \neq |\mathcal{B}_1|$ then return $0$
    $\langle pk,\mathcal{F},b',\{c_j\}_{j=1}^{b}\rangle \leftarrow \mathsf{T}_{\mathsf{t1\text{-}t3}}(\langle\mathcal{F},\mathcal{B}_0\rangle)$
    $\Psi \leftarrow \mathsf{zkpGen}^{\mathsf{hash}}(\langle pk,\{c_j\}_{j=1}^{b}\rangle)$
    $\hat{\mathsf{b}} \leftarrow \mathfrak{M}_2^{\mathsf{hash}}(\langle pk,\mathcal{F},b',\{c_j\}_{j=1}^{b},\Psi\rangle,\phi)$
    return $\hat{\mathsf{b}}$
```

```
Experiment $\mathsf{Expt}^{\mathsf{T},01}(\langle\mathfrak{M}_1,\mathfrak{M}_2\rangle)$
    $\langle\mathcal{B}_0,\mathcal{B}_1,\phi\rangle \leftarrow \mathfrak{M}_1^{\mathsf{zkpSim.hash}}(\mathcal{F})$
    if $|\mathcal{B}_0| \neq |\mathcal{B}_1|$ then return $0$
    $\langle pk,\mathcal{F},b',\{c_j\}_{j=1}^{b}\rangle \leftarrow \mathsf{T}_{\mathsf{t1\text{-}t3}}(\langle\mathcal{F},\mathcal{B}_0\rangle)$
    $\Psi \leftarrow \mathsf{zkpSim.prove}(\langle pk,\{c_j\}_{j=1}^{b}\rangle)$
    $\hat{\mathsf{b}} \leftarrow \mathfrak{M}_2^{\mathsf{zkpSim.hash}}(\langle pk,\mathcal{F},b',\{c_j\}_{j=1}^{b},\Psi\rangle,\phi)$
    return $\hat{\mathsf{b}}$
```

```
Experiment $\mathsf{Expt}^{\mathsf{T},10}(\langle\mathfrak{M}_1,\mathfrak{M}_2\rangle)$
    $\mathsf{hash} \xleftarrow{\$} \mathcal{H}$
    $\langle\mathcal{B}_0,\mathcal{B}_1,\phi\rangle \leftarrow \mathfrak{M}_1^{\mathsf{hash}}(\mathcal{F})$
    if $|\mathcal{B}_0| \neq |\mathcal{B}_1|$ then return $0$
    $\langle pk,\mathcal{F},b',\{c_j\}_{j=1}^{b}\rangle \leftarrow \mathsf{T}_{\mathsf{t1\text{-}t3}}(\langle\mathcal{F},\mathcal{B}_1\rangle)$
    $\Psi \leftarrow \mathsf{zkpGen}^{\mathsf{hash}}(\langle pk,\{c_j\}_{j=1}^{b}\rangle)$
    $\hat{\mathsf{b}} \leftarrow \mathfrak{M}_2^{\mathsf{hash}}(\langle pk,\mathcal{F},b',\{c_j\}_{j=1}^{b},\Psi\rangle,\phi)$
    return $\hat{\mathsf{b}}$
```

```
Experiment $\mathsf{Expt}^{\mathsf{T},11}(\langle\mathfrak{M}_1,\mathfrak{M}_2\rangle)$
    $\langle\mathcal{B}_0,\mathcal{B}_1,\phi\rangle \leftarrow \mathfrak{M}_1^{\mathsf{zkpSim.hash}}(\mathcal{F})$
    if $|\mathcal{B}_0| \neq |\mathcal{B}_1|$ then return $0$
    $\langle pk,\mathcal{F},b',\{c_j\}_{j=1}^{b}\rangle \leftarrow \mathsf{T}_{\mathsf{t1\text{-}t3}}(\langle\mathcal{F},\mathcal{B}_1\rangle)$
    $\Psi \leftarrow \mathsf{zkpSim.prove}(\langle pk,\{c_j\}_{j=1}^{b}\rangle)$
    $\hat{\mathsf{b}} \leftarrow \mathfrak{M}_2^{\mathsf{zkpSim.hash}}(\langle pk,\mathcal{F},b',\{c_j\}_{j=1}^{b},\Psi\rangle,\phi)$
    return $\hat{\mathsf{b}}$
```

Fig. 9: Definition of $\mathsf{Expt}^{\mathsf{T},\alpha\beta}(\mathfrak{M})$ for $\alpha,\beta \in \{0,1\}$

indistinguishable from a real execution in $\mathsf{H}(\mathsf{b}',i^*)$. Since $\mathfrak{D}_{\mathrm{CPA}}$ outputs 0 if $\mathfrak{M}$ outputs 0, we have:

$$
\begin{aligned}
&\mathbb{P}\left(\mathsf{Expt}_{\mathcal{E}}^{\mathrm{CPA},\mathsf{b}'}(\mathfrak{D}_{\mathrm{CPA}}) = 0\right)\\
&= \sum_{i^*=1}^{b} \mathbb{P}\left(i = i^*\right) \cdot \mathbb{P}\left(\mathsf{Expt}_{\mathcal{E}}^{\mathrm{CPA},\mathsf{b}'}(\mathfrak{D}_{\mathrm{CPA}}) = 0 \wedge i = i^*\right)\\
&= \sum_{i^*=1}^{b} \frac{1}{b} \cdot \mathbb{P}\left(\mathsf{H}(\mathsf{b}',i^*) = 0\right) \ .
\end{aligned}
$$

Combining the above, we have:

$$
\begin{aligned}
&\mathsf{Adv}_{\mathcal{E}}^{\mathrm{CPA}}(\mathfrak{D}_{\mathrm{CPA}})\\
&= \mathbb{P}\left(\mathsf{Expt}_{\mathcal{E}}^{\mathrm{CPA},0}(\mathfrak{D}_{\mathrm{CPA}}) = 0\right) - \mathbb{P}\left(\mathsf{Expt}_{\mathcal{E}}^{\mathrm{CPA},1}(\mathfrak{D}_{\mathrm{CPA}}) = 0\right)\\
&\geq \sum_{i^*=1}^{b} \frac{1}{b} \cdot \Big(\mathbb{P}\left(\mathsf{H}(0,i^*) = 0\right) - \mathbb{P}\left(\mathsf{H}(1,i^*) = 0\right)\Big)\\
&\geq \frac{1}{b} \cdot \Big(\mathbb{P}\left(\mathsf{H}(0,b) = 0\right) - \mathbb{P}\left(\mathsf{H}(1,1) = 0\right)\Big)\\
&\quad + \sum_{i^*=2}^{b} \frac{1}{b} \cdot \Big(\mathbb{P}\left(\mathsf{H}(0,i^*-1) = 0\right) - \mathbb{P}\left(\mathsf{H}(1,i^*) = 0\right)\Big)\\
&\geq \frac{1}{b} \cdot \Big(\mathbb{P}\left(\mathsf{H}(0,b) = 0\right) - \mathbb{P}\left(\mathsf{H}(1,1) = 0\right)\Big) + 0\\
&\geq \frac{1}{b} \cdot \Big(\mathbb{P}\left(\mathsf{Expt}^{\mathsf{T},01}(\mathfrak{M}) = 0\right) - \mathbb{P}\left(\mathsf{Expt}^{\mathsf{T},11}(\mathfrak{M}) = 0\right)\Big)\\
&\geq \frac{1}{b} \cdot \Big(\mathcal{P}(0,1) - \mathcal{P}(1,1)\Big) \ .
\end{aligned}
$$

Here $\mathfrak{D}_{\mathrm{CPA}}$ makes one "left-or-right" oracle query in constructing $\{c_j\}_{j=1}^{b}$, and runs in time at most $t + b \cdot t_{\mathrm{ENC}} + q_{\mathrm{RO}} \cdot t_{\mathrm{HASH}}$ which is the time for $\mathfrak{M}$ plus the time to produce b ciphertexts and answer at most q_{RO} random oracle queries from $\mathfrak{M}$.

Justification of (26) and (27): Given an M-adversary $\mathfrak{M} = \langle\mathfrak{M}_1,\mathfrak{M}_2\rangle$ for the experiments $\mathsf{Expt}^{\mathsf{T},0\mathsf{b}''}$, we construct an adversary $\mathfrak{D}_{\mathrm{ZKP}}$ for noninteractive zero-knowledge experiment $\mathsf{Expt}_{\Pi}^{\mathrm{ZKP},\mathsf{b}''}$ defined in Sec. B-A2b. $\mathfrak{D}_{\mathrm{ZKP}}$ first invokes $\mathfrak{M}_1$ with $\mathcal{F}$ and receives two distinct Bloom filters $\langle\mathcal{F},\mathcal{B}_0\rangle$ and $\langle\mathcal{F},\mathcal{B}_1\rangle$ (of equal size, b') from $\mathfrak{M}_1$. $\mathfrak{D}_{\mathrm{ZKP}}$ receives, from either zkpGen (if $\mathsf{b}'' = 0$) or $\mathsf{zkpSim.prove}$ (if $\mathsf{b}'' = 1$), a proof Ψ for the statement that $\{c_j\}_{j=1}^{b} \subseteq C_{pk}(g) \cup C_{pk}(g^{-1})$, as well as pk and $\{c_j\}_{j=1}^{b}$ produced based on $\mathcal{B}_0$ as part of the public information for Ψ. Then $\mathfrak{D}_{\mathrm{ZKP}}$ invokes $\mathfrak{M}_2$ with $\langle pk,\mathcal{F},b',\{c_j\}_{j=1}^{b},\Psi\rangle$ as its expected input. Also, $\mathfrak{D}_{\mathrm{ZKP}}$ uses its random oracle to reply to all random oracle queries by $\mathfrak{M}_2$ to verify Ψ (as in line m1 in Fig. 5). Finally, $\mathfrak{D}_{\mathrm{ZKP}}$ outputs 0 if $\mathfrak{M}_2$ outputs 0. Since the simulation provided by $\mathfrak{D}_{\mathrm{ZKP}}$ to $\mathfrak{M}$ is perfectly indistinguishable from a real execution in $\mathsf{Expt}^{\mathsf{T},0\mathsf{b}''}(\mathfrak{M})$, we have:

$$
\begin{aligned}
&\mathsf{Adv}_{\Pi}^{\mathrm{ZKP}}(\mathfrak{D}_{\mathrm{ZKP}})\\
&= \mathbb{P}\left(\mathsf{Expt}_{\Pi}^{\mathrm{ZKP},1}(\mathfrak{D}_{\mathrm{ZKP}}) = 1\right) - \mathbb{P}\left(\mathsf{Expt}_{\Pi}^{\mathrm{ZKP},0}(\mathfrak{D}_{\mathrm{ZKP}}) = 1\right)\\
&= \mathbb{P}\left(\mathsf{Expt}_{\Pi}^{\mathrm{ZKP},0}(\mathfrak{D}_{\mathrm{ZKP}}) = 0\right) - \mathbb{P}\left(\mathsf{Expt}_{\Pi}^{\mathrm{ZKP},1}(\mathfrak{D}_{\mathrm{ZKP}}) = 0\right)\\
&\geq \mathbb{P}\left(\mathsf{Expt}^{\mathsf{T},00}(\mathfrak{M}) = 0\right) - \mathbb{P}\left(\mathsf{Expt}^{\mathsf{T},01}(\mathfrak{M}) = 0\right)\\
&\geq \mathcal{P}(0,0) - \mathcal{P}(0,1) \ .
\end{aligned}
$$

Similarly, we can construct a $\hat{\mathfrak{D}}_{\mathrm{ZKP}}$ like $\mathfrak{D}_{\mathrm{ZKP}}$, except that it receives from zkpGen (if $\mathsf{b}'' = 0$) or $\mathsf{zkpSim.prove}$ (if $\mathsf{b}'' = 1$) a proof Ψ corresponding to $\mathcal{B}_1$, instead of $\mathcal{B}_0$, and that outputs 1 if $\mathfrak{M}$ outputs 0. So:

$$
\begin{aligned}
&\mathsf{Adv}_{\Pi}^{\mathrm{ZKP}}(\hat{\mathfrak{D}}_{\mathrm{ZKP}})\\
&= \mathbb{P}\left(\mathsf{Expt}_{\Pi}^{\mathrm{ZKP},1}(\hat{\mathfrak{D}}_{\mathrm{ZKP}}) = 1\right) - \mathbb{P}\left(\mathsf{Expt}_{\Pi}^{\mathrm{ZKP},0}(\hat{\mathfrak{D}}_{\mathrm{ZKP}}) = 1\right)\\
&\geq \mathbb{P}\left(\mathsf{Expt}^{\mathsf{T},11}(\mathfrak{M}) = 0\right) - \mathbb{P}\left(\mathsf{Expt}^{\mathsf{T},10}(\mathfrak{M}) = 0\right)\\
&\geq \mathcal{P}(1,1) - \mathcal{P}(1,0) \ .
\end{aligned}
$$

Here $\mathfrak{D}_{\mathrm{ZKP}}$ and $\hat{\mathfrak{D}}_{\mathrm{ZKP}}$ make at most q_{RO} random oracle queries and run in time at most $t + b \cdot t_{\mathrm{ENC}} + q_{\mathrm{RO}} \cdot t_{\mathrm{HASH}}$ which is the time-complexity of $\mathfrak{M}$ plus the time costs of producing b ciphertexts and answering at most q_{RO} random oracle queries from $\mathfrak{M}$. ■