# Honey Encryption: Security Beyond the Brute-Force Bound

Ari Juels[1] and Thomas Ristenpart[2]

[1] ajuels@gmail.com
[2] University of Wisconsin–Madison, rist@cs.wisc.edu

**Abstract.** We introduce *honey encryption* (HE), a simple, general approach to encrypting messages using low min-entropy keys such as passwords. HE is designed to produce a ciphertext which, when decrypted with any of a number of *incorrect* keys, yields plausible-looking but bogus plaintexts called *honey messages*. A key benefit of HE is that it provides security in cases where too little entropy is available to withstand brute-force attacks that try every key; in this sense, HE provides security beyond conventional brute-force bounds. HE can also provide a hedge against partial disclosure of high min-entropy keys.
HE significantly improves security in a number of practical settings. To showcase this improvement, we build concrete HE schemes for password-based encryption of RSA secret keys and credit card numbers. The key challenges are development of appropriate instances of a new type of randomized message encoding scheme called a *distribution-transforming encoder* (DTE), and analyses of the expected maximum loading of bins in various kinds of balls-and-bins games.

## 1 Introduction

Many real-world systems rely for encryption on low-entropy or weak secrets, most commonly user-chosen passwords. Password-based encryption (PBE), however, has a fundamental limitation: users routinely pick poor passwords. Existing PBE mechanisms attempt to strengthen bad passwords via salting, which slows attacks against multiple users, and iterated application of one-way functions, which slows decryption and thus attacks by a constant factor $c$ (e.g., $c = 10{,}000$). Recent results [6] prove that for conventional PBE schemes (e.g., [32]), work $q$ suffices to crack a single ciphertext with probability $q/c2^{\mu}$ for passwords selected from a distribution with min-entropy $\mu$. This *brute-force bound* is the best possible for in-use schemes.

Unfortunately empirical studies show this level of security to frequently be insufficient. A recent study [12] reports $\mu < 7$ for passwords observed in a real-world population of 69+ million users. (1.08% of users chose the same password.) For any slowdown $c$ small enough to support timely decryption in normal use, the security offered by conventional PBE is clearly too small to prevent message-recovery (MR) attacks.

We explore a new approach to PBE that provides security beyond the brute-force bound. The idea is to build schemes for which attackers are *unable to succeed in message recovery even after trying every possible password / key*. We formalize this approach by way of a new cryptographic primitive called *honey encryption* (HE). We

provide a framework for realizing HE schemes and show scenarios useful in practice in which even computationally unbounded attackers can provably recover an HE-encrypted plaintext with probability at most $2^{-\mu} + \epsilon$ for negligible $\epsilon$. Since there exists a trivial, fast attack that succeeds with probability $2^{-\mu}$ (guess the most probable password), we thus demonstrate that HE can yield optimal security.

While HE is particularly useful for password-based encryption (PBE), we emphasize that "password" here is meant very loosely. HE is applicable to *any* distribution of low min-entropy keys, including passwords, PINs, biometrically extracted keys, etc. It can also serve usefully as a hedge against partial compromise of high min-entropy keys.

**Background.** Stepping back, let us review briefly how brute-force message-recovery attacks work. Given an encryption $C = \mathsf{enc}(K, M)$ of message $M$, where $K$ and $M$ are drawn from known distributions, an attacker's goal is to recover $M$. The attacker decrypts $C$ under as many candidate keys as she can, yielding messages $M_1, \ldots, M_q$. Should one of the candidate keys be correct (i.e., $K$ is from a low-entropy distribution), $M$ is guaranteed to appear in this list, and at this stage the attacker wins with probability equal to her ability to pick out $M$ from the $q$ candidates. Conventional PBE schemes make this easy in almost all settings. For example, if $M$ is a 16-digit credit card number encoded via ASCII and the PBE scheme acts like an ideal cipher, the probability that any $M_i \neq M$ is a valid ASCII encoding of a 16-digit string is negligible, at $(10/256)^{16} < 2^{-74}$. An attacker can thus reject incorrect messages and recover $M$ with overwhelming probability. In fact, cryptographers generally ignore the problem of identifying valid plaintexts and assume conservatively that if $M$ appears in the list, the attacker wins.

Prior theoretical frameworks for analyzing PBE schemes have focused on showing strong security bounds for sufficiently unpredictable keys. Bellare, Ristenpart, and Tessaro [6] prove of PKCS#5 PBE schemes that no attacker can break semantic security (learn partial information about plaintexts) with probability greater than $q/(c2^{\mu})$; here, $c$ is the time to perform a single decryption, $\mu$ is the min-entropy of the distribution of the keys, and negligible terms are ignored. As mentioned above, though, when $\mu = 7$, such a result provides unsatisfying security guarantees, and the formalisms and proof techniques of [6] cannot offer better results. It may seem that this is the best one can do and that providing security beyond this "brute-force barrier" remains out of reach.

Perhaps unintuitively (at least to the authors of the present paper), the bounds above are actually *not* tight for all settings, as they do not take into account the distribution of the challenge message $M$. Should $M$ be a uniformly chosen bit-string of length longer than $\mu$, for instance, then the best possible message recovery attack would appear to work with probability at most $1/2^{\mu}$. This is because for typical PBE schemes an attacker will have a hard time, in practice, distinguishing the result of $\mathsf{dec}(K, C)$ for any $K$ from a uniform bit string. Said another way, the candidate messages $M_1, \ldots, M_q$ would all appear to be equally valid as plaintexts. Thus an adversary would seem to maximize her probability of message recovery simply by decrypting $C$ using the key with the highest probability, which is at most $1/2^{\mu}$.

Previously proposed security tools have exploited exactly this intuition for special cases. Hoover and Kausik [26] consider the problem of encrypting a (uniformly-chosen) RSA or DSA secret exponent for authenticating a user to a remote system. Only the remote system holds the associated public key. To hedge against compro-

mise of the user's machine, they suggest encrypting the secret exponent under a PIN (a short decimal-string password). They informally argue that brute-force decryption yields valid-looking exponents, and that an attacker can at best use each candidate exponent in a brute-force online attack against the remote system. Their work led to a commercially deployed system [29]. Other systems similarly seek to foil offline brute-force attacks, but mainly by means of hiding valid authentication credentials in an *explicitly stored list* of plausible-looking fake ones (often called "decoys" or "honeywords") [10, 28]. Similarly, detection of system breaches using "honeytokens," such as fake credit-card numbers, is a common industry practice [38].

**Honey encryption (HE).** Inspired by such decoy systems, we set out to build HE schemes that provide security beyond the brute-force barrier. These schemes yield candidate messages during brute-force attacks that are indistinguishable from valid ones. We refer to the incorrect plaintext candidates in HE as *honey messages*, following the long established role of this sweet substance in computer security terminology.

We provide a formal treatment of HE. Functionally, an HE scheme is exactly like a PBE scheme: it takes arbitrary strings as passwords and uses them to perform randomized encryption of a message. We ask that HE schemes simultaneously target two security goals: message recovery (MR) security, as parameterized by a distribution over messages, and the more (multi-instance) semantic-security style goals of [6]. As we noted, the latter can only be achieved up to the brute-force barrier, and is thus meaningful only for high min-entropy keys; our HR schemes achieve the goals of [6] using standard techniques. The bulk of our efforts in this paper will be on MR security, where we target security better than $q/c2^{\mu}$. Our schemes will, in fact, achieve security bounds close to $1/2^{\mu}$ for unbounded attackers when messages are sufficiently unpredictable.

HE schemes can also produce compact ciphertexts (unlike explicitly stored decoys). While lengths vary by construction and message distribution, we are able to give schemes for which the HE ciphertext for $M$ can be as small as a constant multiple (e.g., 2) of the length of a conventional PBE ciphertext on $M$.

**Framework for HE schemes.** We provide a general methodology for building HE schemes. Its cornerstone is a new kind of (randomized) message encoding that we call a *distribution-transforming encoder (DTE)*. A DTE is designed with an estimate of the message distribution $p_m$ in mind, making it conceptually similar to arithmetic/Huffman coding [19]. The message space for a DTE is exactly the support of $p_m$ (messages with non-zero probability). Encoding a message sampled from $p_m$ yields a "seed" value distributed (approximately) uniformly. It is often convenient for seeds to be binary strings. A DTE must have an efficient decoder that, given a seed, obtains the corresponding message. Applying the decoder to a uniformly sampled seed produces a message distributed (approximately) under $p_m$. A good (secure) DTE is such that no attacker can distinguish with significant probability between these two distributions: (1) a pair $(M, S)$ generated by selecting $M$ from $p_m$ and encoding it to obtain seed $S$, and (2) a pair $(M, S)$ generated by selecting a seed $S$ uniformly at random and decoding it to obtain message $M$. Building DTEs is non-trivial in many cases, for example when $p_m$ is non-uniform.

Encrypting a message $M$ under HE involves a two-step procedure that we call *DTE-then-encrypt*. First, the DTE is applied to $M$ to obtain a seed $S$. Second, the seed $S$ is

encrypted under a conventional encryption scheme $\mathsf{enc}$ using the key $K$, yielding an HE ciphertext $C$. This conventional encryption scheme $\mathsf{enc}$ must have message space equal to the seed space and all ciphertexts must decrypt under any key to a valid seed. Typical PBE schemes operating on bitstrings provide all of this (but authenticated encryption schemes do not). Appropriate care must be taken, however, to craft a DTE whose outputs require no padding (e.g., for CBC-mode encryption).

We prove a general theorem (Theorem 2) that upper bounds the MR security of any DTE-then-encrypt scheme by the DTE's security and a scheme-specific value that we call the expected maximum load. Informally, the expected maximum load measures the worst-case ability of an unbounded attacker to output the right message; we relate it to the expected maximum load of a bin in a kind of balls-and-bins game. Analyzing an HE scheme built with our approach (and a good DTE) therefore reduces to analyzing the balls-and-bins game that arises for the particular key and message distribution. Assuming the random oracle model or ideal cipher model for the underlying conventional encryption scheme enables us to assume balls are thrown independently in these games. (We conjecture that $k$-wise independent hashing, and thus $k$-wise independent ball placement, may achieve strong security in many cases as well.)

A DTE is designed using an estimate of the target message distribution $p_m$. If the DTE is only approximately right, we can nevertheless prove message-recovery security far beyond the brute-force-barrier. If the DTE is bad, i.e., based on a poor estimate of $p_m$, we fall back to normal security (up to the brute-force barrier), at least provably achieving the semantic security goals in [6]. This means we never do worse than prior PBE schemes, and, in particular, attackers must always first perform the work of offline brute-force attacks before HE security becomes relevant.

**HE instantiations.** We offer as examples several concrete instantiations of our general DTE-then-encrypt construction. We build HE schemes that work for RSA secret keys by crafting a DTE for uniformly chosen pairs of prime numbers. This enables us to apply HE to RSA secret keys as used by common tools such as OpenSSL, and improves on the non-standard selection of RSA secret exponents in Hoover and Kausik [26]. Interestingly, simple encoding strategies here fail. For example, encoding the secret keys directly as binary integers (in the appropriate range) would enable an attacker to rule out candidate messages resulting from decryption by running primality tests. Indeed, the DTE we design has decode (essentially) implement a prime number generation algorithm. (This approach slows down decryption significantly, but as noted above, in PBE settings slow decryption can be advantageous.)

We also build HE schemes for password-based encryption of credit card numbers, their associated Card Verification Values (CVVs), and (user-selected) PINs. Encryption of PINs requires a DTE that handles a non-uniform distribution over messages, as empirical studies show a heavy user bias in PIN selection [8]. The resulting analysis consequently involves a balls-and-bins game with non-uniform bin capacities, a somewhat unusual setup in the literature.

In each of the cases above we are able to prove close to optimal MR security.

**Limitations of HE.** The security guarantees offered by HE come with some strings attached. First, HE security does not hold when the adversary has side information about

the target message. As a concrete example, the RSA secret key HE scheme provides strong MR guarantees only when the attacker does not know the public key associated with the encrypted secret key. Thus the HE cannot effectively protect normal HTTPS certificate keys. (The intended application for this HE scheme is client authorization, where the public key is stored only at the remote server, a typical setting for SSH users. See, e.g., [26].) Second, because decryption of an HE ciphertext under a wrong key produces fake but valid-looking messages, typos in passwords might confuse legitimate users in some settings. We address this issue of "typo-safety" in Section 7. Third and finally, we assume in our HE analyses that the key and message distributions are independent. If they are correlated, an attacker may be able to identify a correct message by comparing it with the decryption key that produced it. Similarly, encrypting two correlated messages under the same key may enable an adversary to identify correct messages. (Encrypting independent messages under the same key is fine.) We emphasize, however, that should any of these assumptions fail, HE security falls back to normal PBE security: there is never any harm in using HE.

**Full version.** Due to page constraints, this abstract omits proofs and some other content. Refer to the full version of the paper for the omitted material [27].

## 2 Related Work

Our HE schemes provide a form of information-theoretic encryption, as their MR security does not rely on any computational hardness assumption. Information-theoretic encryption schemes, starting with the one-time pad [36], have seen extensive study. Most closely related is entropic security [21, 35], where the idea is to exploit high-entropy messages to perform encryption that leaks no predicate on the plaintext even against unbounded attackers (and hence beyond the brute-force bound). Their goal was to enable use of uniform, smaller (than one-time pads) keys yet achieve information-theoretic security. HE similarly exploits the entropy of messages, but also provides useful bounds (by targeting MR security) even when the combined entropy of messages and keys is insufficient to achieve entropic security. See also the discussion in the full version.

Deterministic [2, 4, 11] and hedged [3, 34] public-key encryption rely on entropy in messages to offset having no or only poor randomness during encryption. HE similarly exploits adversarial uncertainty about messages in the case that keys are poor; HE can be viewed as "hedging" against poor keys (passwords) as opposed to poor randomness.

In natural applications of HE, the message space $\mathcal{M}$ must encompass messages of special format, rather than just bitstrings. In this sense, HE is related to format-preserving encryption (FPE) [5], although HE is randomized and has no preservation requirement (our ciphertexts are unstructured bit strings). An implication of our approach, however, is that some FPE constructions (e.g., for credit-card encryption) can be shown to achieve HE-like security guarantees when message distributions are uniform. HE is also conceptually related to collisionful hashing [9], the idea of creating password hashes for which it is relatively easy to find inverses and thus hard to identify the original, correct password (as opposed to identifying a correct message).

Under (non-interactive) non-committing encryption [17, 31], a ciphertext can be "opened" to an arbitrary message under a suitably selected key. (For example, a one-

time pad is non-committing.) HE has a different requirement, namely that decrypting a fixed ciphertext under different keys yields independent-looking samples of the message space. Note that unlike non-committing encryption [31], HE is achievable in the non-programmable random oracle model. Deniable encryption [16] also allows ciphertexts to be opened to chosen messages; HE schemes do not in general offer deniability.

Canetti, Halevi, and Steiner [18] propose a protocol in which a password specifies a subset of CAPTCHAs that must be solved to decrypt a credential store. Their scheme creates ambiguity around where human effort can be most effectively invested, rather than around the correctness of the contents of the credential store, as HE would.

Perhaps most closely related to HE is a rich literature on deception and decoys in computer security. Honeypots, fake computer systems intended to attract and study attacks, are a stock-in-trade of computer security research [37]. Researchers have proposed honeytokens [20, 38], which are data objects whose use signals a compromise, and honeywords [28], a system that uses passwords as honeytokens. Additional proposals include false documents [14], false network traffic [13], and many variants.

The Kamouflage system [10] is particularly relevant. It conceals a true password vault encrypted under a true master password among $N$ bogus vaults encrypted under bogus master passwords. Kamouflage requires $O(N)$ storage. With a suitable DTE, HE can in principle achieve similar functionality and security with $O(1)$ storage. Kamouflage and related systems require the construction of plausible decoys. This problem has seen study specifically for password protection in, e.g., [10, 28], but to the best of our knowledge, we are the first to formalize it with the concept of DTEs.

## 3 HE Overview

**HE schemes.** An HE scheme has syntax and semantics equivalent to that of a symmetric encryption scheme. Encryption maps a key and message to a ciphertext and, in our schemes, is randomized. Decryption recovers messages from ciphertexts. The departure from conventional symmetric encryption schemes will be in how HE decryption behaves when one uses the wrong key in attempting to decrypt a ciphertext. Instead of giving rise to some error, decryption will emit a plaintext that "looks" plausible.

Formally, let $\mathcal{K}$ and $\mathcal{M}$ be sets, the key space and message space. For generality, we assume that $\mathcal{K}$ consists of variable-length bit strings. (This supports, in particular, varying length passwords.) An HE scheme $\mathsf{HE} = (\mathsf{HEnc}, \mathsf{HDec})$ is a a pair of algorithms. Encryption $\mathsf{HEnc}$ takes input a key $K \in \mathcal{K}$, message $M \in \mathcal{M}$, some uniform random bits, and outputs a ciphertext $C$. We write this as $C \leftarrow\!\!\$\ \mathsf{HEnc}_K(M)$, where $\leftarrow\!\!\$$ denotes that $\mathsf{HEnc}$ may use some number of uniform random bits. Decryption $\mathsf{HDec}$ takes as input a key $K \in \mathcal{K}$, ciphertext $C$, and outputs a message $M \in \mathcal{M}$. Decryption, always deterministic, is written as $M \leftarrow \mathsf{HDec}_K(C)$.

We require that decryption succeeds: Formally, $\Pr[\mathsf{HDec}_K(\mathsf{HEnc}_K(M)) = M] = 1$ for all $K \in \mathcal{K}$ and $M \in \mathcal{M}$, where the event is defined over the randomness in $\mathsf{HEnc}$.

We will write $\mathsf{SE} = (\mathsf{enc}, \mathsf{dec})$ to denote a conventional symmetric encryption scheme, but note that the syntax and semantics match those of an HE scheme.

**Message and key distributions.** We denote a distribution on set $S$ by a map $p\colon S \to [0, 1]$ and require that $\sum_{s \in S} p(s) = 1$. The min-entropy of a distribution is defined to be

$-\log \max_{s \in S} p(s)$. Sampling according to such a distribution is written $s \leftarrow_p S$, and we assume all sampling is efficient. We use $p_m$ to denote a message distribution over $\mathcal{M}$ and $p_k$ for a key distribution over $\mathcal{K}$. Thus sampling according to these distributions is denoted $M \leftarrow_{p_m} \mathcal{M}$ and $K \leftarrow_{p_k} \mathcal{K}$. Note that we assume that draws from $p_m$ and $p_k$ are independent, which is not always the case but will be in our example applications; see Section 7. Whether HE schemes can provide security for any kind of dependent distributions is an interesting question for future work.

**Message recovery security.** To formalize our security goals, we use the notion of security against message recovery attacks. Normally, one aims that, given the encryption of a message, the probability of any adversary recovering the correct message is negligible. But this is only possible when both messages and keys have high entropy, and here we may have neither. Nevertheless, we can measure the message recovery advantage of any adversary concretely, and will do so to show (say) that attackers cannot achieve advantage better than $1/2^{\mu}$ where $\mu$ is the min-entropy of the key distribution $p_k$.

Formally, we define the MR security game as shown in Figure 1 and define advantage for an adversary $\mathcal{A}$ against a scheme $\mathsf{HE}$ by $\mathbf{Adv}^{\mathrm{mr}}_{\mathsf{HE},p_m,p_k}(\mathcal{A}) = \Pr[\mathrm{MR}^{\mathcal{A}}_{\mathsf{HE},p_m,p_k} \Rightarrow \mathsf{true}]$. When working in the random oracle (RO) model, the MR game additionally has a procedure implementing a random function that $\mathcal{A}$ may query. For our schemes, we allow $\mathcal{A}$ to run for an unbounded amount of time and make an unbounded number of queries to the RO. For simplicity we assume $p_m$ and $p_k$ are independent of the RO.

$\underline{\mathrm{MR}^{\mathcal{A}}_{\mathsf{HE},p_m,p_k}}$
$K^* \leftarrow_{p_k} \mathcal{K}$
$M^* \leftarrow_{p_m} \mathcal{M}$
$C^* \leftarrow\!\!\$\ \mathsf{HEnc}(K^*, M^*)$
$M \leftarrow\!\!\$\ \mathcal{A}(C^*)$
return $M = M^*$

**Fig. 1.** Game defining MR security.

**Semantic security.** In the case that keys are sufficiently unpredictable and adversaries are computationally bounded, our HE schemes will achieve semantic security [24]. Our schemes will therefore never provide worse confidentiality than conventional encryption, and in particular the MR advantage in this case equals the min-entropy of the message distribution $p_m$ plus the (assumed) negligible semantic security term. When combined with a suitable password-based key-derivation function [32], our schemes will also achieve the multi-instance security guarantees often desired for password-based encryption [6]. Note that the results in [6] still hold only for attackers that cannot exhaust the min-entropy of the key space.

In the full version we discuss why existing or naïve approaches, e.g., conventional encryption or hiding a true plaintext in a list of fake ones, aren't satisfactory HE schemes.

## 4 Distribution-Transforming Encoders

We introduce a new type of message encoding scheme that we refer to as a *distribution-transforming encoder* (DTE). Formally, it is a pair $\mathsf{DTE} = (\mathsf{encode}, \mathsf{decode})$ of algorithms. The usually randomized algorithm $\mathsf{encode}$ takes as input a message $M \in \mathcal{M}$ and outputs a value in a set $\mathcal{S}$. We call the range $\mathcal{S}$ the *seed space* for reasons that will become clear in a moment. The deterministic algorithm $\mathsf{decode}$ takes as input a value

$S \in \mathcal{S}$ and outputs a message $M \in \mathcal{M}$. We call a DTE scheme *correct* if for any $M \in \mathcal{M}$, $\Pr[\mathsf{decode}(\mathsf{encode}(M)) = M] = 1$.

A DTE encodes a priori knowledge of the message distribution $p_m$. One goal in constructing a DTE is that $\mathsf{decode}$ applied to uniform points provides sampling close to that of a target distribution $p_m$. For a given DTE (that will later always be clear from context), we define $p_d$ to be the distribution over $\mathcal{M}$ defined by

$$p_d(M) = \Pr\left[\, M' = M \;:\; U \leftarrow\!\!\$\, \mathcal{S} \;;\; M' \leftarrow \mathsf{decode}(S) \,\right] .$$

We will often refer to $p_d$ as the DTE distribution. Intuitively, in a good or secure DTE, the distributions $p_m$ and $p_d$ are "close."

Formally, we define this notion of DTE security or goodness, as follows. Let $\mathcal{A}$ be an adversary attempting to distinguish between the two games shown in Figure 2. We define advantage of an adversary $\mathcal{A}$ for a message distribution $p_m$ and encoding scheme $\mathsf{DTE} = (\mathsf{encode}, \mathsf{decode})$ by

$$\mathbf{Adv}^{\mathrm{dte}}_{\mathsf{DTE},p_m}(\mathcal{A}) = \left|\Pr\left[\,\mathrm{SAMP1}^{\mathcal{A}}_{\mathsf{DTE},p_m} \Rightarrow 1\,\right] - \Pr\left[\,\mathrm{SAMP0}^{\mathcal{A}}_{\mathsf{DTE}} \Rightarrow 1\,\right]\right| .$$

While we focus mostly on adversaries with unbounded running times, we note that these measures can capture computationally-good DTEs as well. A perfectly secure DTE is a scheme for which the indistinguishability advantage is zero for even unbounded adversaries. In the full version we explore another way of measuring DTE goodness that, while more complex, sometimes provides slightly better bounds.

**Inverse sampling DTE.** We first build a general purpose DTE using inverse sampling, a common technique for converting uniform random variables into ones from some other distribution. Let $F_m$ be the cumulative distribution function (CDF) associated with a known message distribution $p_m$ according to some ordering of $\mathcal{M} = \{M_1, \ldots, M_{|\mathcal{M}|}\}$. Define $F_m(M_0) = 0$. Let the seed space be $\mathcal{S} = [0, 1)$. Inverse sampling picks a value according to $p_m$ by selecting $S \leftarrow\!\!\$\, [0, 1)$; it outputs $M_i$ such that $F_m(M_{i-1}) \le S < F_m(M_i)$. This amounts to computing the inverse CDF $M = F_m^{-1}(S) = \min_i\{F_m(M_i) > S\}$. The associated DTE scheme $\mathsf{IS\text{-}DTE} = (\mathsf{is\text{-}encode}, \mathsf{is\text{-}decode})$ encodes by picking uniformly from the range $[F_m(M_{i-1}), F_m(M_i))$ for input message $M_i$, and decodes by computing $F_m^{-1}(S)$.

| $\underline{\mathrm{SAMP1}^{\mathcal{B}}_{\mathsf{DTE},p_m}}$ | $\underline{\mathrm{SAMP0}^{\mathcal{B}}_{\mathsf{DTE}}}$ |
|---|---|
| $M^* \leftarrow_{p_m} \mathcal{M}$ | $S^* \leftarrow\!\!\$\, \mathcal{S}$ |
| $S^* \leftarrow\!\!\$\, \mathsf{encode}(M^*)$ | $M^* \leftarrow \mathsf{decode}(S^*)$ |
| $b \leftarrow\!\!\$\, \mathcal{B}(S^*, M^*)$ | $b \leftarrow\!\!\$\, \mathcal{B}(S^*, M^*)$ |
| return $b$ | return $b$ |

**Fig. 2.** Games defining DTE goodness.

All that remains is to fix a suitably granular representation of the reals between $[0, 1)$. The representation error gives an upper bound on the DTE security of the scheme. We defer the details and analysis to the full version. Encoding and decoding each work in time $\mathcal{O}(\log |\mathcal{M}|)$ using a tables of size $\mathcal{O}(|\mathcal{M}|)$, though its performance can easily be improved for many special cases (e.g., uniform distributions).

**DTEs for RSA secret keys.** We turn to building a DTE for RSA secret keys. A popular key generation algorithm generates an RSA key of bit-length $2\ell$ via rejection sampling

of random values $p, q \in [2^{\ell-1}, 2^{\ell})$. The rejection criterion for either $p$ or $q$ is failure of a Miller-Rabin primality test [30, 33]; the resulting distribution of primes is (essentially) uniform over the range. The private exponent is computed as $d = e^{-1} \bmod (p-1)(q-1)$ for some fixed $e$ (typically $65537$), yielding secret key $(N, d)$ and public key $(N, e)$. Usually, the key $p, q$ is stored with some ancillary values (not efficiently recoverable from $d$) to speed up exponentiation via the Chinese Remainder Theorem. Since for fixed $e$, the pair $p, q$ fully defines the secret key, we now focus on building DTEs that take as input primes $p, q \in [2^{\ell-1}, 2^{\ell})$ for some $\ell$ and aim to match the message distribution $p_m$ that is uniformly distributed over the primes in $[2^{\ell-1}, 2^{\ell})$.

One strawman approach is just to encode the input $p, q$ as a pair of $(\ell - 2)$-bit strings (the leading '1' bit left implicit), but this gives a poor DTE. The prime number theorem indicates that an $\ell$-bit integer will be prime with probability about $1/\ell$; thus an adversary $\mathcal{A}$ that applies primality tests to a candidate plaintext has a (very high) DTE advantage of about $1 - 1/\ell^2$.

We can instead adapt the rejection-sampling approach to prime generation itself as a DTE, RSA-REJ-DTE $=$ (rsa-rej-encode, rsa-rej-decode), which works as follows. Encoding (rsa-rej-encode) takes a pair of primes $(p, q)$, constructs a vector of $t$ bitstrings selected uniformly at random from the range $[2^{\ell-1}, 2^{\ell})$, replaces the first (resp. second) prime integer in the list by $p$ (resp. $q$), and outputs the modified vector of $t$ integers (each encoded using $\ell - 2$ bits). (If there's one prime and it's not the last integer in the vector, then that prime is replaced by $p$ and the last integer is replaced by $q$. Should there be no primes in the vector, or one prime in the last position, then the last two integers in the vector are replaced by $(p, q)$.) Decoding (rsa-rej-decode) takes as input a vector of the $t$ integers, and outputs its first two primes. If there do not exist two primes, then it outputs some (hard-coded) fixed primes.[3] For simplicity, we assume a perfect primality testing algorithm; it is not hard to generalize to probabilistic ones.[4] We obtain the following security bound.

**Theorem 1.** *Let $p_m$ be uniform over primes in $[2^{\ell-1}, 2^{\ell})$ for some $\ell \geq 2$ and let RSA-REJ-DTE be the scheme described above. Then $\mathbf{Adv}^{\mathrm{dte}}_{\textsf{RSA-REJ-DTE},p_m}(\mathcal{A}) \leq (1 - 1/(3\ell))^{t-1}$ for any adversary $\mathcal{A}$.*

This scheme is simple, but a small adversarial advantage does translate into a large encoding. For example with $\ell = 1024$ (2048-bit RSA), in order to achieve a bound of $\mathbf{Adv}^{\mathrm{dte}}_{\textsf{RSA-REJ-DTE},p_m}(\mathcal{A}) < 10^{-5}$ requires $t \geq 35{,}361$, resulting in an encoding of about 4.5 megabytes. (Assuming keys of low entropy, $10^{-5}$ is small enough to contribute insignificantly to security bounds on the order of those in Section 7.) It may be tempting to try to save on space by treating $S$ as a seed for a pseudorandom generator (PRG) that is then used to generate the $t$ values during decoding. Encoding, though, would then need to identify seed values that map to particular messages (prime pairs), effectively inverting the PRG, which is infeasible.

Some RSA key generators do not use rejection-sampling, but instead use the classic algorithm that picks a random integer in $[2^{\ell-1}, 2^{\ell})$ and increments it by two until a

[3] We could also output bottom, but would then need to permit errors in decoding and HE decryption.

[4] Doing so would also require our definition of DTE correctness to allow errors.

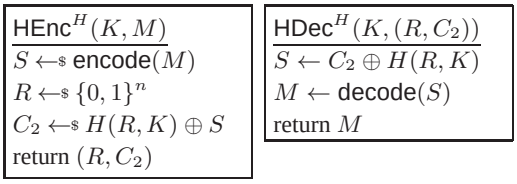


**Fig. 3.** A particularly simple instantiation of DTE-then-Encrypt using a hash-function $H$ to implement the symmetric encryption.

prime is found (c.f., [15,25]). In this case, a DTE can be constructed (see the full version for details) that requires only $2(\ell - 2)$-bit seeds, and so is space-optimal. Other, more randomness-efficient rejection-sampling techniques [23] may also be used to obtain smaller encodings.

In some special settings it may be possible to hook existing key-generation software, extract the PRG key / seed $\kappa$ used for the initial generation of an RSA key pair, and apply HE directly to $\kappa$. A good DTE (and thus HE scheme) can then be constructed trivially, as $\kappa$ is just a short (e.g., 256-bit) uniformly random bitstring.

## 5 DTE-then-Encrypt Constructions

We now present a general construction for HE schemes for a target distribution $p_m$. Intuitively, the goal of any HE scheme is to ensure that the plaintext resulting from decrypting a ciphertext string under a key is indistinguishable from freshly sampling a plaintext according to $p_m$. Let $\mathsf{DTE} = (\mathsf{encode}, \mathsf{decode})$ be a DTE scheme whose outputs are in the space $\mathcal{S} = \{0,1\}^s$. Let $\mathsf{SE} = (\mathsf{enc}, \mathsf{dec})$ be a conventional symmetric encryption scheme with message space $\mathcal{S}$ and some ciphertext space $\mathcal{C}$.

Then DTE-then-Encrypt $\mathsf{HE}[\mathsf{DTE}, \mathsf{SE}] = (\mathsf{HEnc}, \mathsf{HDec})$ applies the DTE encoding first, and then performs encryption under the key. Decryption works in the natural way. It is easy to see that the resulting scheme is secure in the sense of semantic security (when keys are drawn from a large enough space) should SE enjoy the same property.

We fix a simple instantiation using a hash function $H : \{0,1\}^n \times \mathcal{K} \to \mathcal{S}$ to perform symmetric encryption, see Figure 3. It is denoted as $\mathsf{HE}[\mathsf{DTE}, H]$. Of course, one should apply a password-based key-derivation function to $K$ first, as per [32]; we omit this for simplicity.

To analyze security, we use the following approach. First we establish a general theorem (Theorem 2) that uses the goodness of the DTE scheme to move to a setting where, intuitively, the attacker's best bet is to output the message $M$ that maximizes the probability (over choice of key) of $M$ being the result of decrypting a random challenge ciphertext. The attacker wins, then, with exactly the sum of the probabilities of the keys that map the ciphertext to that message. Second, we define a weighted balls-and-bins game with non-uniform bin sizes in a way that makes the expected load of the maximally loaded bin at the end of the game exactly the winning probability of the attacker. We can then analyze these balls-and-bins games for various message and key distributions combinations (in the random oracle model). We put all of this together

to derive bounds for some concrete applications in Section 7, but emphasize that the results here provide a general framework for analyzing HE constructions.

**Applying DTE goodness.** Let $\mathcal{K}_{M,C} = \{K \ : \ K \in \mathcal{K} \wedge M = \mathsf{HDec}(K, C)\}$ be the set of keys that decrypt a specific ciphertext to a specific message and (overloading notation slightly) let $p_k(\mathcal{K}_{M,C}) = \sum_{K \in \mathcal{K}_{M,C}} p_k(K)$ be the aggregate probability of selecting a key that falls in any such set. Then for any $C \in \mathcal{C}$ we define $L_{\mathsf{HE},p_k}(C) = \max_M p_k(\mathcal{K}_{M,C})$. Let $L_{\mathsf{HE},p_k}$ represent the random variable $L_{\mathsf{HE},p_k}(C)$ defined over $C$ uniformly chosen from $\mathcal{C}$ and any coins used to define $\mathsf{HDec}$. (For example in the hash-based scheme, we take this over the coins used to define $H$ when modeled as a random oracle.) We will later show, for specific message/key distributions and using balls-and-bins-style arguments, bounds on $\mathrm{E}[\, L_{\mathsf{HE},p_k}\,]$. We call this value the expected maximum load, following the terminology from the balls-and-bins literature.

For the following theorem we require from $\mathsf{SE}$ only that encrypting uniform messages gives uniform ciphertexts. More precisely, that $S \leftarrow\!\!\$\ \mathcal{S}\ ;\ C \leftarrow\!\!\$\ \mathsf{enc}(K, S)$ and $C \leftarrow\!\!\$\ \mathcal{C}\ ;\ S \leftarrow \mathsf{dec}(K, C)$ define identical distributions for any key $K \in \mathcal{K}$. This is true for many conventional schemes, including the hash-based scheme used in Figure 3, CTR mode over a block-cipher, and CBC-mode over a block cipher (assuming the DTE is designed so that $\mathcal{S}$ includes only bit strings of length a multiple of the block size). The proof of the following theorem is given in the full version.

**Theorem 2.** *Fix distributions $p_m, p_k$, an encoding scheme $\mathsf{DTE}$ for $p_m$, and a symmetric encryption scheme $\mathsf{SE} = (\mathsf{enc}, \mathsf{dec})$. Let $\mathcal{A}$ be an MR adversary. Then we give a specific adversary $\mathcal{B}$ in the proof such that $\mathbf{Adv}^{\mathrm{mr}}_{\mathsf{HE},p_m,p_k}(\mathcal{A}) \ \leq \ \mathbf{Adv}^{\mathrm{dte}}_{\mathsf{DTE},p_m}(\mathcal{B}) + \mathrm{E}[\, L_{\mathsf{HE},p_k}\,]$. Adversary $\mathcal{B}$ runs in time that of $\mathcal{A}$ plus the time of one $\mathsf{enc}$ operation.*

**The balls-and-bins interpretation.** What remains is to bound $\mathrm{E}[\, L_{\mathsf{HE},p_k}\,]$. To do so, we use the following equivalent description of the probability space as a type of balls-and-bins game. Uniformly pick a ciphertext $C \leftarrow\!\!\$\ \mathcal{C}$. Each ball represents one key $K$ and has weight equal to $p_k(K)$. We let $a = |\mathcal{K}|$ be the number of balls. Each bin represents a message $M$ and $b = |\mathcal{M}|$ is the number of bins.[5] A ball is placed in a particular bin should $C$ decrypt under $K$ to the message labeling that bin. Then $L_{\mathsf{HE},p_k}$ as defined above is exactly the random variable defined as the maximum, over bins, sum of weights of all balls thrown into that bin. In this balls-and-bins game the balls are weighted, the bins have varying capacities, and the (in)dependence of ball throws depends on the details of the symmetric encryption scheme used.

To derive bounds, then, we must analyze the expected maximum load for various balls-and-bins games. For brevity in the following sections we focus on the hash-based HE scheme shown in Figure 3. By modeling $H$ as a random oracle,[6] we get that all the ball throws are independent. At this stage we can also abstract away the details of the DTE, instead focusing on the distribution $p_d$ defined over $\mathcal{M}$. The balls-and-bins game is now completely characterized by $p_k$ and $p_d$, and we define the random variable $L_{p_k,p_d}$ as the load of the maximally loaded bin at the end of the balls-and-bins game that

[5] Convention is to have $m$ balls and $n$ bins, but we use $a$ balls and $b$ bins to avoid confusion since $m$ connotes messages.

[6] Technically speaking we only require the non-programmable random oracle [22, 31].

throws $|\mathcal{K}|$ balls with weights described by $p_k$ independently into $|\mathcal{M}|$ bins, choosing a bin according to $p_d$. The following lemma formalizes this transition.

**Lemma 1.** *Consider* $\mathsf{HE}[\mathsf{DTE}, H]$ *for* $H$ *modeled as a RO and* $\mathsf{DTE}$ *having distribution* $p_d$*. For any key distribution* $p_k$*,* $\mathrm{E}[L_{\mathsf{HE},p_k}] \leq \mathrm{E}[L_{p_k,p_d}]$.

We give similar lemmas for block-cipher based modes (in the ideal cipher model) in the full version. Thus we can interchange the hash-based symmetric encryption scheme for other ones in the final results of Section 7 with essentially the same security bounds.

## 6 Balls-and-Bins Analyses

In this section we derive bounds for various types of balls-and-bins games, as motivated and used for the example applications of HE in the next section. These cases are by no means exhaustive; they illustrate the power of our general HE analysis framework. Treating $p_k$ and $p_d$ as vectors, we can write their dimension as $|p_k| = a$ and $|p_d| = b$.

In the special case of $a = b$ and both $p_k$ and $p_d$ uniform, the balls-and-bins game becomes the standard one. One can use the classic proof to show that $\mathrm{E}[\, L_{p_k,p_d}\,] \leq \frac{1}{b} + \frac{3 \ln b}{b \ln \ln b}$. HE schemes for real applications, however, are unlikely to coincide with this special case, and so we seek other bounds.

**Majorization.** To analyze more general settings, we exploit a result due to Berenrink, Friedetzky, Hu, and Martin [7] that builds on a technique called "majorization" earlier used for the balls-and-bins setting by Azar, Broder, Karlin, and Upfal [1].

Distributions such as $p_k$ and $p_d$ can be viewed as vectors of appropriate dimension over $\mathbb{R}$. We assume below that vector components are in decreasing order, e.g. that $p_k(i) \geq p_k(j)$ for $i < j$. Let $m$ be a number and $p_k, p'_k \in \mathbb{R}^a$. Then $p'_k$ *majorizes* $p_k$, denoted $p'_k \succ p_k$, if $\sum_{i=1}^{a} p'_k[i] = \sum_{i=1}^{a} p_k[i]$ and $\sum_{i=1}^{j} p'_k[i] \geq \sum_{i=1}^{j} p_k[i]$ for all $1 \leq j \leq a$.

Majorization intuitively states that $p'_k$ is more "concentrated" than $p_k$: a prefix of any length of $p'_k$ has cumulative weight at least as large as the cumulative weight of the same-length prefix of $p_k$. We have the following theorem from [7, Cor. 3.5], slightly recast to use our terminology. We also extend our definition of load to include the $i$ highest loaded bins: let $L^i_{p_k,p_d}$ be the random variable which is the total weight in the $i$ highest-loaded bins at the end of the balls-and-bins game.

**Theorem 3 (BFHM08).** *Let* $p_k, p'_k, p_d$ *be distributions. If* $p'_k \succ p_k$*, then* $\mathrm{E}[L^i_{p'_k,p_d}] \geq \mathrm{E}[L^i_{p_k,p_d}]$ *for all* $i \in [1, b]$.

Consider the case $i = 1$, which corresponds to the expected maximum bin loads for the two key distributions. As a concrete example, let $p_k = (1/2, 1/4, 1/4)$, $p'_k = (1/2, 1/2, 0)$. Then $p'_k \succ p_k$ and thus $\mathrm{E}\,[\, L(p'_k, p_d)\,] \geq \mathrm{E}\,[\, L(p_k, p_d)\,]$ because "fusion" of the two 1/4-weight balls into one ball biases the expected maximum load upwards.

Our results will use majorization to shift from a setting with non-uniform key distribution $p_k$ having max-weight $w$ to a setting with uniform key distribution with weight $\lceil 1/w \rceil$.

**Non-uniform key distributions.** We turn now to giving a bound for the case that $p_k$ has maximum weight $w$ (meaning $p_k(M) \leq w$ for all $M$) and $p_d$ is uniform. In our

examples in the next section we have that $a \ll b$, and so we focus on results for this case. We start with the following lemma (whose proof is given in the full version).

**Lemma 2.** *Suppose $p_k$ has maximum weight $w$ and $p_d$ is such that $b = ca$ for some positive integer $c$. Then for any positive integer $s > 2e/c$, where $e$ is Euler's constant, it holds that*

$$\mathrm{E}\left[\, L_{p_k,p_d} \,\right] \leq w \left( (s-1) + 2 \left( \frac{a^2}{c^{s-1}} \right) \left( \frac{e}{s} \right)^s \right) .$$

For cases in which $b = \mathcal{O}(a^2)$, a convenient, somewhat tighter bound on $\mathrm{E}\left[\, L_{p_k,p_d} \,\right]$ is possible. We observe that in many cases of interest, the term $r(c,b)$ in the bound below will be negligible. Proof of this next lemma is given in the full version.

**Lemma 3.** *Suppose $p_k$ has maximum weight $w$ and $p_d$ is such that $b = ca^2$ for some positive integer $c$. Then $\mathrm{E}\left[\, L_{p_k,p_d} \,\right] \leq w \left[ 1 + \frac{1}{2c} + r(c,b) \right]$, where $e$ is Euler's constant and $r(c,b) = \left( \frac{e}{27c^2} \right) \left( 1 - \frac{e}{cb} \right)^{-1}$.*

**Non-uniform balls-and-bins.** To support our examples in the next section, we also consider the case of non-uniform $p_d$ . Proof of this lemma is given in the full version.

**Lemma 4.** *Let $L_{\mathcal{B}}$ denote the maximum load yielded by throwing $a$ balls (of weight 1) into a set $\mathcal{B}$ of $b$ bins of non-uniform capacity at most $0 \leq \gamma \leq 3 - \sqrt{5}$. Let $L_{\mathcal{B}^*}$ denote the maximum load yielded by throwing $a^* = 3a$ balls (of weight 1) into a set $\mathcal{B}^*$ of $b^* = \lfloor 2/\gamma \rfloor$ bins of uniform capacity. Then $\mathrm{E}[L_{\mathcal{B}}] \leq \mathrm{E}[L_{\mathcal{B}^*}]$.*

## 7 Example Applications, Bounds, and Deployment Considerations

We now draw together the results of the previous sections into some concrete examples involving honey encryption of RSA secret keys and credit card data. For concreteness, we assume password-based encryption of these secrets, although our proven results are much more general. Appealing again to Bonneau's Yahoo! study [12] in which the most common password was selected by $1.08\% \approx 1/100$ of users, we assume for simplicity that the maximum-weight password / key is selected with probability $w = 1/100$. (At this level of entropy, prior security results for PBE schemes are not very useful.)

### 7.1 HE for Credit Card Numbers, PINs, and CVVs

We first consider application of HE to credit card numbers. For convenience, we evaluate HE as applied to a single value, e.g., one credit-card number. Recall, though, that HE security is unaffected by simultaneous encryption of multiple, independent messages drawn from the same distribution. So our security bounds in principle apply equally well to encryption of a vault or repository of multiple credit-card numbers.

A (Mastercard or Visa) credit card number, known technically as a Primary Account Number (PAN), consists of sixteen decimal digits. Although structures vary somewhat, commonly nine digits constitute the cardholder's account number, and may be regarded

as selected uniformly at random upon issuance. One digit is a (mod 10) checksum (known as the Luhn formula). A useful result then is the following theorem, whose proof is given in the full version.

**Theorem 4.** *Consider* $\mathsf{HE}[\mathsf{IS\text{-}DTE}, H]$ *with* $H$ *modeled as a RO and* IS-DTE *using an* $\ell$*-bit representation. Let* $p_m$ *be a uniform distribution over* $b$ *messages and* $p_k$ *be a key-distribution with maximum weight* $w$*. Let* $\alpha = \lceil 1/w \rceil$*. Then for any adversary* $\mathcal{A}$*,* $\mathbf{Adv}^{\mathrm{mr}}_{\mathsf{HE},p_m,p_k}(\mathcal{A}) \leq w(1+\delta) + \dfrac{1+\alpha}{2^\ell}$ *where* $\delta = \frac{\alpha^2}{2b} + \frac{e\alpha^4}{27b^2}\left(1 - \frac{e\alpha^2}{b^2}\right)^{-1}$.

For many cases of interest, $b \gg \alpha^2$, and thus $\delta$ will be small. We can also set $\ell$ appropriately to make $(1+\alpha)/2^\ell$ negligible. Theorem 4 then yields a simple and useful bound, as for our next two examples.

As cardholder account numbers are uniformly selected nine-digit values, they induce a uniform distribution over a space of $b = 10^9$ messages. Given $w = 1/100$, then, $\alpha^2/b = 10^{-5}$ and so $\delta \approx 0$. The upper bound on MR advantage is $w = 1/100$. This bound is essentially tight, as there exists an adversary $\mathcal{A}$ achieving advantage $w = \frac{1}{100}$. Namely, the adversary that decrypts the challenge ciphertext with the most probable key and then outputs the resulting message. This adversary has advantage at least $w$.

Finally, consider encrypting both 5-digits of the credit-card / debit-card account number (the last 4 digits still considered public) along with the user's PIN number. (Credit card PINs are used for cash withdrawals and to authorize debit-card transactions.) A detailed examination of a corpus of 3.4 million user-selected PINs is given in [8], and gives in particular a CDF that can be used to define an inverse sampling DTE. The most common user-selected PIN is '1234'; it has an observed frequency of 10.713%. Thus, PINs have very little minimum entropy (roughly 3 bits). Combining a PIN with a five-digit effective account number induces a *non-uniform* message space, with maximum message probability $\gamma = 1.0713 \times 10^{-6}$. Consequently, Theorem 4 is not applicable to this example.

A variant of the proof of Theorem 4, however, that makes use of Lemma 4 for non-uniform bin sizes, establishes the following corollary.

**Corollary 1.** *Consider* $\mathsf{HE}[\mathsf{IS\text{-}DTE}, H]$ *with* $H$ *modeled as a RO and* IS-DTE *using an* $\ell$*-bit representation. Let* $p_m$ *be a non-uniform distribution with maximum message probability* $\gamma \leq 3 - \sqrt{5}$*, and* $p_k$ *be a key-distribution with maximum weight* $w$*. Let* $\alpha = \lceil 1/w \rceil$*. Then for any adversary* $\mathcal{A}$*,* $\mathbf{Adv}^{\mathrm{mr}}_{\mathsf{HE},p_m,p_k}(\mathcal{A}) \leq w(1+\delta) + \dfrac{(1+\alpha)}{2^\ell}$ *where* $\delta = \frac{\overline{\alpha}^2}{2\overline{b}} + \frac{e\overline{\alpha}^4}{27\overline{b}^2}\left(1 - \frac{e\overline{\alpha}^2}{\overline{b}^2}\right)^{-1}$ *and* $\overline{\alpha} = \lceil 3/w \rceil$ *and* $\overline{b} = \lfloor 2/\gamma \rfloor$.

Corollary 1 yields a bound defined by the expected maximum load of a balls-and-bins experiment with 300 balls (of weight $w = 1/100$) and $\lfloor 2/\gamma \rfloor = 1{,}866{,}890$ uniform-capacity bins, with $c = \overline{\alpha}^2/\overline{b} = 1/20.74$. The final MR bound is therefore about $1.02\%$. This is slightly better than the bound of the previous example (at $1.05\%$). It shows, significantly, that Corollary 1 is tight enough to give improved bounds despite the scant minimum entropy in a PIN.

Credit cards often have an associated three- or four-digit *card verification value*, a secret used to conduct transactions. In the full version, we investigate the case of applying HE to such small messages.

### 7.2 HE for RSA Secret Keys

We now show how to apply HE to RSA secret keys using the DTE introduced for this purpose in Section 4.

In some settings, RSA is used without making a user's public key readily available to attackers. A common example is RSA-based client authentication to authorize access to a remote service using HTTPS or SSH. The client stores an RSA secret / private key and registers the corresponding public key with the remote service.

Practitioners recommend encrypting the client's secret key under a password to provide defense-in-depth should the client's system be passively compromised.[7] With password-based encryption, though, an attacker can mount an offline brute-force attack against the encrypted secret key. Use of straightforward unauthenticated encryption wouldn't help here: as the secret key is usually stored as a pair of primes $p$ and $q$ (to facilitate use of the Chinese Remainder Theorem), an attacker can quickly test the correctness of a candidate secret key by applying a primality test to its factors. Similarly, given the passwords used in practice (e.g., for $w = 1/100$), key-hardening mechanisms (e.g., iterative hashing) do not provide an effective slowdown against brute-force attack. Cracking a password-encrypted RSA secret key remains fairly easy.

HE is an attractive option in this setting. To build an HE scheme for $2\ell$-bit RSA secret keys we can use the DTE from Section 4. We have the following theorem.

**Theorem 5.** *Consider* $\mathsf{HE}[\mathsf{RSA\text{-}REJ\text{-}DTE}, H]$ *with* $\mathsf{RSA\text{-}REJ\text{-}DTE}$ *the* $2\ell$*-bit RSA DTE using seed space vectors of size* $t$ *and* $H$ *modeled as a RO. Let* $p_m$ *be uniform over primes in* $[2^{-\ell-1}, 2^{\ell})$ *and let* $p_k$ *be a key-distribution with maximum weight* $w$*. Let* $\alpha = \lceil 1/w \rceil$*. Then for any adversary* $\mathcal{A}$ *it holds that*

$$\mathbf{Adv}^{\mathrm{mr}}_{\mathsf{HE},p_m,p_k}(\mathcal{A}) \leq w(1+\delta) + (1+\alpha)\left(1 - \frac{1}{3\ell}\right)^{t-1}$$

*where* $\delta = \frac{\alpha^2}{2\lceil 2^{\ell-1}/\ell \rceil} + \left(\frac{e\alpha^4}{27\lceil 2^{\ell-1}/\ell \rceil^2}\right) \cdot \left(1 - \frac{e\alpha^2}{\lceil 2^{\ell-1}/\ell \rceil^2}\right)^{-1}$ .

The proof is much like that of Theorem 4 (the full version): apply Theorem 2; plug in the advantage upper bound for the RSA rejection sampling DTE (Theorem 1); apply Lemma 1 to get independent ball tosses; majorize to get uniform-weighted balls (Theorem 3); apply a union bound to move from $p_d$ back to uniform bin selection; and then finally apply the balls-and-bins analysis for uniform bins (Lemma 3).

The term $\delta$ is small when $-\log w \ll \ell$. For example, with $\ell = 1024$ and $w = 1/100$ and setting $t = 35{,}393$, we have that $\delta \approx 0$ and the overall MR advantage is upper bounded by $1.1\%$. The ciphertext size will still be somewhat large, at about 4.5 megabytes; one might use instead the DTEs discussed in the full version for which similar MR bounds can be derived yet ciphertext size ends up short.

[7] Obviously an active attacker can sniff the keyboard or otherwise capture the secret key. We also are ignoring the role of network attackers that may also gain access to transcripts dependent on the true secret key. See [26] for discussion.

### 7.3 Deployment considerations

A number of considerations and design options arise in the implementation and use of HE. Here we briefly mention a couple involving the use of checksums.

**Typo-safety.** Decryption of an HE ciphertext $C^*$ under an incorrect password / key $K$ yields a fake but valid-looking message $M$. This is good for security, but can be bad for usability if a fake plaintext appears valid to a legitimate user.

One possible remedy, proposed in [28], is the use of error-detecting codes or checksums, such as those for ISBN book codes. For example, a checksum on the password / key $K^*$ might be stored with the ciphertext $C^*$. Such checksums would reduce the size of the key space $\mathcal{K}$ and cause some security degradation, and thus require careful construction and application. Another option in some cases is online verification of plaintexts. For example, if a credit-card number is rejected by an online service after decryption, the user might be prompted to re-enter her password.

**Honeytokens without explicit sharing.** In [10], it is suggested that fake passwords / honeytokens be shared explicitly between password vault applications and service providers. Application of error-correcting codes to plaintexts in HE can create *honeytokens without explicit sharing*. As a naïve example (and crude error-correcting code), an HE scheme for credit-card numbers might explicitly store the first two digits of the credit-card account number. If a service provider then receives an invalid credit-card number in which these digits are correct, it gains evidence of a decryption attempt on the HE ciphertext by an adversary. This approach degrades security slightly by reducing the message space, and must be applied with care. But it offers an interesting way of coupling HE security with online security checks.

## 8 Conclusion

Low-entropy secrets such as passwords are likely to persist in computer systems for many years. Their use in encryption leaves resources vulnerable to offline attack. Honey encryption can offer valuable additional protection in such scenarios. HE yields plausible looking plaintexts under decryption with invalid keys (passwords), so that offline decryption attempts alone are insufficient to discover the correct plaintext. HE also offers a gracefully degrading hedge against partial disclosure of high min-entropy keys, and, by simultaneously meeting standard PBE security notions should keys be high entropy, HE never provides worse security than existing PBE schemes.

We showed applications in which HE security upper bounds are equal to an adversary's conditional knowledge of the key distribution, i.e., they min-entropy of keys. These settings have message space entropy greater than the entropy of keys, but our framework can also be used to analyze other settings.

A key challenge for HE—as with all schemes involving decoys—is the generation of plausible honey messages through good DTE construction. We have described good DTEs for several natural problems. For the case where plaintexts consist of passwords, e.g., password vaults, the relationship between password-cracking and DTE construction mentioned above deserves further exploration. DTEs offer an intriguing way of

potentially repurposing improvements in cracking technology to achieve improvements in encryption security by way of HE.

More generally, for human-generated messages (password vaults, e-mail, etc.), estimation of message distributions via DTEs is interesting as a natural language processing problem. Similarly, the reduction of security bounds in HE to the expected maximum load for balls-and-bins problems offers an interesting connection with combinatorics. The concrete bounds we present can undoubtedly be tightened for a variety of cases. Finally, a natural question to pursue is what kinds of HE bounds can be realized in the standard model via, e.g., $k$-wise independent hashing.

## Acknowledgements

The authors thank the anonymous reviewers, as well as Daniel Wichs and Mihir Bellare, for their insightful comments.

## References

1. Y. Azar, A. Broder, A. Karlin, and E. Upfal. Balanced allocations. *SIAM journal on computing*, 29(1):180–200, 1999.
2. M. Bellare, A. Boldyreva, and A. ONeill. Deterministic and efficiently searchable encryption. In *Advances in Cryptology – CRYPTO 2007*, pages 535–552. Springer Berlin Heidelberg, 2007.
3. M. Bellare, Z. Brakerski, M. Naor, T. Ristenpart, G. Segev, H. Shacham, and S. Yilek. Hedged public-key encryption: How to protect against bad randomness. In *Advances in Cryptology – ASIACRYPT 2009*, pages 232–249. Springer Berlin Heidelberg, 2009.
4. M. Bellare, M. Fischlin, A. ONeill, and T. Ristenpart. Deterministic encryption: Definitional equivalences and constructions without random oracles. In *Advances in Cryptology – CRYPTO 2008*, pages 360–378. Springer Berlin Heidelberg, 2008.
5. M. Bellare, T. Ristenpart, P. Rogaway, and T. Stegers. Format-preserving encryption. In *Selected Areas in Cryptography*, pages 295–312, 2009.
6. M. Bellare, T. Ristenpart, and S. Tessaro. Multi-instance security and its application to password-based cryptography. In *Advances in Cryptology – CRYPTO 2012*, pages 312–329. Springer Berlin Heidelberg, 2012.
7. P. Berenbrink, T. Friedetzky, Z. Hu, and R. Martin. On weighted balls-into-bins games. *Theoretical Computer Science*, 409(3):511 – 520, 2008.
8. N. Berry. PIN analysis. DataGenetics blog, 2012.
9. T. A. Berson, L. Gong, and T.M.A. Lomas. Secure, keyed, and collisionful hash functions. Technical Report SRI-CSL-94-08, SRI International Laboratory, 1993 (revised 2 Sept. 1994).
10. H. Bojinov, E. Bursztein, X. Boyen, and D. Boneh. Kamouflage: loss-resistant password management. In *ESORICS*, pages 286–302, 2010.
11. A. Boldyreva, S. Fehr, and A. ONeill. On notions of security for deterministic encryption, and efficient constructions without random oracles. In *Advances in Cryptology – CRYPTO 2008*, pages 335–359. Springer Berlin Heidelberg, 2008.
12. J. Bonneau. The science of guessing: analyzing an anonymized corpus of 70 million passwords. In *IEEE Symposium on Security and Privacy*, pages 538–552. IEEE, 2012.
13. B. M. Bowen, V. P. Kemerlis, P. Prabhu, A. D. Keromytis, and S. J. Stolfo. Automating the injection of believable decoys to detect snooping. In *WiSec*, pages 81–86. ACM, 2010.

14. B.M. Bowen, S. Hershkop, A. D. Keromytis, and S. J. Stolfo. *Baiting Inside Attackers Using Decoy Documents*, pages 51–70. 2009.
15. J. Brandt and I. Damgård. On generation of probable primes by incremental search. In *Advances in Cryptology – Crypto 1992*, pages 358–370. Springer, 1993.
16. R. Canetti, C. Dwork, M. Naor, and R. Ostrovsky. Deniable encryption. In *Advances in Cryptology – CRYPTO 1997*, pages 90–104. Springer, 1997.
17. R. Canetti, U. Friege, O. Goldreich, and M. Naor. Adaptively secure multi-party computation. 1996.
18. R. Canetti, S. Halevi, and M. Steiner. Hardness amplification of weakly verifiable puzzles. In *TCC*, pages 17–33, 2005.
19. T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*, pages 428–436. MIT Press, third edition, 2009.
20. A. Paes de Barros. IDS mailing list, "RES: Protocol anomaly detection IDS – honeypots". `http://seclists.org/focus-ids/2003/Feb/95`, Feb. 2003.
21. Y. Dodis and A. Smith. Entropic security and the encryption of high entropy messages. In *Theory of Cryptography Conference (TCC)*, pages 556–577, 2005.
22. M. Fischlin, A. Lehmann, T. Ristenpart, T. Shrimpton, M. Stam, and S. Tessaro. Random oracles with (out) programmability. In *Advances in Cryptology – ASIACRYPT 2010*, pages 303–320. Springer Berlin Heidelberg, 2010.
23. P.A. Fouque and M. Tibouchi. Close to uniform prime number generation with fewer random bits. Cryptology ePrint Archive, Report 2011/481, 2011. `http://eprint.iacr.org/`.
24. S. Goldwasser and S. Micali. Probabilistic encryption. *Journal of computer and system sciences*, 28(2):270–299, 1984.
25. J. Gordon. Strong primes are easy to find. In *Advances in Cryptology – Eurocrypt 1984*, pages 216–223. Springer, 1985.
26. D.N. Hoover and B.N. Kausik. Software smart cards via cryptographic camouflage. In *IEEE Symposium on Security and Privacy*, pages 208–215. IEEE, 1999.
27. A. Juels and T. Ristenpart. Honey encryption: Beyond the brute-force barrier (full version), 2014. `http://pages.cs.wisc.edu/~rist/papers/honeyenc.html`.
28. A. Juels and R. Rivest. Honeywords: Making password-cracking detectable. In *ACM Conference on Computer and Communications Security – CCS 2013*, pages 145–160. ACM, 2013.
29. B. Kausik. Method and apparatus for cryptographically camouflaged cryptographic key. U.S. Patent 6,170,058, 2001.
30. G. Miller. Riemann's hypothesis and tests for primality. *Journal of computer and system sciences*, 13(3):300–317, 1976.
31. J.B. Nielsen. Separating random oracle proofs from complexity theoretic proofs: The non-committing encryption case. In *Advances in Cryptology – Crypto 2002*, pages 111–126. Springer, 2002.
32. PKCS #5: Password-based cryptography standard (rfc 2898). RSA Data Security, Inc., September 2000. Version 2.0.
33. M. Rabin. Probabilistic algorithms. *Algorithms and Complexity*, 21, 1976.
34. T. Ristenpart and S. Yilek. When good randomness goes bad: Virtual machine reset vulnerabilities and hedging deployed cryptography. In *NDSS*, 2010.
35. A. Russell and H. Wang. How to fool an unbounded adversary with a short key. In *Advances in Cryptology – EUROCRYPT 2002*, pages 133–148, 2002.
36. C.E. Shannon. Communication theory of secrecy systems. *Bell System Technical Journal*, 28(4):656–715, 1948.
37. L. Spitzner. *Honeypots: Tracking Hackers*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2002.
38. L. Spitzner. Honeytokens: The other honeypot. Symantec SecurityFocus, July 2003.