# Bounds on surmising remixed keys

Daniel R. L. Brown*

Rough draft
April 20, 2015

**Abstract**

A remixed key is derived from a secret source key by applying a public but unpredictable random function to the source key. A remixed key models a key derived from a shared secret and a public unpredictable salt, using a common, deterministic, pseudorandom function—which is somewhat like TLS record-layer keys.

This report tries to validate the intuition that remixed keys are not easy to surmise, in other words, that remixing does not introduce an exploitable spike in the probability distribution of the remixed key. The report provides pencil-and-paper proofs of numerical bounds on the probability that an adversary can surmise a remixed key, assuming a uniformly random source key and remix function. The proofs are derived from a proof of an asymptotic result on probability theory in a textbook by Shoup.

## 1 Introduction

Let $R$ and $S$ be finite non-empty sets and let $R^S$ be the set of functions from $S$ to $R$. Let $R^{R^S}$ be the set of functions from $R^S$ to $R$. Let the *surmisability* (of remixing $S$ to $R$) be the number:

$$\mu = \max_{e \in R^{R^S}} \Pr\left[(s, f) \in_\$ S \times R^S : e(f) = f(s)\right], \tag{1}$$

where the notation $u \in_\$ U$ indicates a random element $u$ uniformly distributed in set $U$, and the notation $U \times V$ indicates the Cartesian product of sets consisting of all pairs $(u, v)$ with $u \in U$ and $v \in V$.

The surmisability $\mu$ depends only on the size of $R$ and $S$. As a concrete example, suppose $S$ has size $|S| = 2^{192}$ and $R$ has size $|R| = 2^{128}$. How easily can a useful upper bound like $\mu \leq 2^{-50}$ be proved?

*dbrown@certicom.com (Certicom/BlackBerry research/standards)

### 1.1 Theoretical cryptographic application

The surmisability $\mu$ is the maximum possible success rate of an adversary Eve in the following theoretical cryptographic game.

1. Alice and/or Bob securely establish a secret $s \in_\$ S$ and a random function $f \in_\$ R^S$.
2. Alice and/or Bob compute $r = f(s)$, which can be called a *remixed* key.
3. The function $f$ is not kept secret, so subsequently revealed to the adversary Eve.
4. Eve makes a surmisal $r' \in R$ based on $f$.
5. Eve wins if $r' = r$, in other words, Eve wins if she correctly surmises Alice and/or Bob's remixed key.

Eve's success rate in this game is defined prior to the choice of $f$ made by Alice and/or Bob. Of course, in each instance of the game, after $f$ is revealed to Eve, Eve's probability of success may vary depending upon $f$, which we can write as $\mu|_f$, and call the $f$-conditional surmisability. Note that $\mu = E_f(\mu|_f)$, so surmisability is the average $f$-surmisability.

Suppose that Eve has a maximal success rate but uses a probabilistic algorithm. If some non-negligible subset of Eve's random choices would result in a lower success rate, then Eve could drop these choices and improve her success rate. Therefore, essentially all of Eve's random choices must lead to the same success rate. So, in theory, Eve can use a deterministic algorithm of the same success rate. Eve effectively selects a deterministic function $e \in R^{R^S}$ that maps the function $f$ to her surmise $r'$. Eve succeeds when $e(f) = r' = r = f(s)$. Hence definition (1).

To consider the maximum possible success rate, we grant Eve unlimited computational power during the phase in which she can make her surmisal. Granting Eve this power serves mainly to avoid making computational hardness assumptions about the functions modeled by the random function $f$.

Note that it is not the intention of the remixed model to let Eve retain unlimited computational power before and after the remixing step. In many real-world cases, Alice and/or Bob will use $r$ before Eve needs to surmise $r$, in which provides Eve the additional opportunities to test numerous guesses for $r$. The surmisability does not quantify Eve's multiple-guessing (searching) attacks after Eve witnesses usage of $r$, but intuitively there seems less reason

to worry that remixing significantly worsens such guessing attacks. (See §B.1 for some definitions and further informal discussion of this setting.)

In short, the question of usefully bounding surmisability seems like such a basic and simple question, that it is worth answering. The motivation for the remixed key model and the surmisability is further discussed in §A.

### 1.2 Other security issues related to remixed keys

A low surmisability is potentially important for security, but it is not necessarily sufficient for remixed key to be secure. Other important security questions can be asked for remixed keys. Remixed keys are not completely realistic, and other variants of remixed keys may be important to consider. Some of these questions and variants are discussed briefly in §B.

### 1.3 Previous work: Shoup's result

This report's upper bounds and proofs are adapted from a result and its one-page proof by Shoup [Sho09, Theorem 8.29] about the balls-and-bins problem (the expected maximum number of balls in any one bin after random tossing). Shoup's theorem is rigorous but asymptotic. Consequently, it leaves open for us the question of whether the $o(1)$ term will be small enough to offer any security for 128-bit keys. This report takes some small steps further in the proofs to remove the ambiguity arising from asymptoticity.

Shoup's result corresponds to case $|R| = |S|$ where the remixed and source key-space have the same size. This report aims to be more general in handling arbitrary sizes for $R$ and $S$.

Furthermore, this report also tries to keep its proofs simple, in the sense of using fewer proof tools where possible. For example, the report separates out some special choices of key-space sizes, in which the proofs in this report can avoid basic calculus, or even avoid using logarithms.

This report is much longer than Shoup's, partly because of the reasons above, but also because many of the proofs have been divided into many small steps. The larger number of small step are a tad tedious to verify, but I needed them for debugging the frequent mistakes I kept making (such as reversing $\leq$ and $\geq$.) Another style quirk in many proofs are the end-to-end chains of inequalities (instead of separately bounded sub-quantities being re-assembled into the final inequality). These chains helped me keep sight of the big picture, but give longer proofs because some formulas repeat in these chains. In short, I tried to keep the proofs simple, rather than short.

Other similar previous work is discussed in §C.

## 2 Basic lower bounds

The basic (and effectively well-known) lower bounds are:

$$\frac{1}{|R|}, \frac{1}{|S|} \leq \mu \tag{2}$$

To prove the first basic lower bound, choose some $r' \in_\$ R$, and let Eve always surmise $r'$ as her guess, regardless of the function $f$. In other words, Eve ignores the remix function. The probability that $r = r'$ is $1/|R|$. In terms of Eve's function $e$: choose $r' \in_\$ R$, and let $e(f) = r'$ for all $f$, so $e$ is constant with value $r'$.

Actually, since the remix function is random, Eve can pick any fixed $r'$ she likes, not necessarily a random $r'$. Eve's strategy of using a random $r'$ is more general in the sense that it also works against non-random remix functions.

To prove the second basic lower bound, consider the adversary Eve who choose $s' \in_\$ S$ and guesses $r' = f(s')$. The probability that $s' = s$ is at least $1/|S|$. If $s' = s$, then $r' = r$. In terms of Eve's function $e$: choose $s' \in_\$ S$ and let $e(f) = f(s')$. (So, $e$ is non-constant: it varies with $f$).

The lower bound $1/|S| \leq \mu$ formalizes the idea that a remixed key has no more secrecy than the source key, since no new secrecy was added. The lower bound $1/|R| \leq \mu$ formalizes the idea that a remixed key can be no better than a key freshly generated in the remix key-space.

Unifying these two lower bounds gives:

$$\max\left(|R|^{-1}, |S|^{-1}\right) \leq \mu. \tag{3}$$

All these lower bounds do not provide any security protection, but they do show some of the theoretical limits on security on the remixed keys. Any upper bounds must obviously exceed these lower bounds.

## 3 Discrete upper bounds

The following discrete upper bounds are given two essentially equivalent proofs with different notation.

**Theorem 1.** *Let $m = |S|$ and $n = |R|$ and $p$ be a non-negative integer. Then*

$$\mu \leq u(p) = \frac{pn^m + \binom{m}{p+1} n^{m-p}}{mn^m}. \tag{4}$$

*Proof described using games and adversaries.* The desired upper bound for Eve's success rate is $\mu \leq \frac{p}{m} + \frac{1}{m}\binom{m}{p+1}n^{-p}$. Suppose Eve encounters function $f$, and computes $r' = e(f)$ as her surmisal for $r = f(s)$. Consider two cases based on how $s$ relates to Eve's surmisal.

In the first case, $s$ is among the first $p$ elements of the set $f^{-1}(r')$ (in some ordering of $S$). Since $s$ was chosen independently of $f$, the probability that $s$ is among the first $p$ elements of $f^{-1}(r')$ is at most $p/m$.

In the second case, $s$ is not among the first $p$ elements of the set $f^{-1}(r')$. If Eve is successful then $s$ is not among the first $p$ elements of $f^{-1}(f(s))$. We need to bound the probability of this event.

Let $T$ the set of $p$ elements of $f^{-1}(f(s)) < s$. For any given set $T$ of $p$ elements of $S \setminus s$, the probability that $f(t) = f(s)$ for all $t \in T$ is $n^{-p}$. For $s$ and $T$ drawn uniformly at random and independently, the probability that $s$ larger than all the element of $T$ is $\frac{1}{p+1}$, because the index of $s$ in $s \cup T$ is uniformly distributed. Summing probabilities over all possible $(s, T)$ gives $\binom{m-1}{p}\frac{1}{p+1}n^{-p} = \binom{m}{p+1}n^{-p}$. □

*Proof described using sets and functions.* Fix $e \in R^{R^S}$ as the function maximizing the probability in the surmisability. Hence $|f^{-1}(e(f))| \geq |f^{-1}(r)|$ for all $r \in R$. Let $E$ be the set (event):

$$E = \{(s, f) \in S \times R^S : e(f) = f(s)\} \tag{5}$$

that defines the surmisability probability: so $\mu = |E|/|S \times R^S| = \frac{|E|}{mn^m}$.

Equip $S$ with some arbitrary order, allowing elements of $S$ to be sorted and indexed. If $s \in T$ and $T$ is an ordered set, let $i(s, T)$ be the index of $s$ in $T$, meaning $i(s, T) = 1$ if $s$ is the smallest element of $T$, and so on. If $s \notin T$, then $i(s, T)$ is undefined. If $T \subseteq U$, let $i^+(s, T, U)$ be the index of $s$ in $U$ re-ordered so that elements $T$ are considered first. For example, if $s$ is the smallest element of $U \setminus T$, then $i(s, T, U) = |T| + 1$. In particular, if $i \in T$, then $i^+(s, T, U) = i(s, T)$.

For convenience, any inequality involving an undefined value of $i(s, T)$ or $i(s, T, U)$ is considered false. Let:

$$D = \{(s, f) : i(s, f^{-1}(e(f))) \leq p\}, \tag{6}$$

$$B = \{(s, f) : i(s, f^{-1}(e(f))) > p\}, \tag{7}$$

which are disjoint sets with $E = D \cup B$. To see this: re-write the condition for $E$ that $f(s) = e(f)$ as $s \in f^{-1}(e(f))$; then partition $E$ according the index.

Next, let

$$D^+ = \{(s, f) : i^+(s, f^{-1}(e(f)), S) \leq p\}, \tag{8}$$
$$B^+ = \{(s, f) : i(s, f^{-1}(f(s))) > p\}, \tag{9}$$

We claim that $D \subseteq D^+$ and $B \subseteq B^+$. If $(s, f) \in D$, then $s \in f^{-1}(e(f))$, so $i^+(s, f^{-1}(e(f)), S) = i(s, f^{-1}(e(f))) \leq p$, and thus $(s, f) \in D^+$. If $(s, f) \in B$, then $(s, f) \in E$, so $e(f) = f(s)$, and $i(s, f^{-1}(f(s))) = i(s, f^{-1}(e(f))) > p$, and thus $(s, f) \in B^+$. Therefore:

$$|E| = |D| + |B| \leq |D^+| + |B^+|. \tag{10}$$

Consider $D^+$. For every $s \in S$, the index $i^+(s, f^{-1}(e(f)), S)$ is always defined, and at most $m$. Furthermore, for a fixed $f$, each value of the index determines a unique $s$. Therefore, $|D^+| = pn^m$ if $p \leq m$, and $|D^+| \leq pn^m$ in general.

Consider $B^+$. For a given $(s, f) \in B^+$, let $T = \{t : i(t, f^{-1}(f(s)) \leq p + 1\}$. Then $T$ has $p + 1$ elements, because $i(s, f^{-1}(f(s))) \geq p + 1$. Note that $f(t) = f(s)$ for all $t \in T$. Let $S'$ be obtained from $S$ by identifying all the elements of $T$. Then $|S'| = m - p$. Let $f'$ be the function from $S'$ to $R$ corresponding to $f$. Then $(s, f)$ determines a pair $(T, f')$. Furthermore, $(T, f')$ determines $(s, f)$, so $(T, f')$ is unique to $(s, f)$. Hence $|B^+|$ is at most the number of such pairs, which is $\binom{m}{p+1}n^{m-p}$. □

The upper bounds $u(0) = u(m) = 1$ are vacuous: as weak as possible as an upper bound for an adversary's success rate. For $p > m$, the bound $u(p) > 1$ is even worse. Therefore, the only possible useful bounds $u(p)$ are such that $1 \leq p < m$.

All the of the remaining results in this report hinge on the upper bounds $u(p)$, so we fix the notation henceforth in this report.

Because of the lower bound (3) on $\mu$, the next corollary follows immediately, but, just to test against the possibility that the previous theorem is flagrantly wrong, we provide an independent proof of the corollary.

**Corollary 1.** *For all integers $p \geq 0$ and $m, n \geq 1$:*

$$\max\left(\tfrac{1}{m}, \tfrac{1}{n}\right) \leq u(p). \tag{11}$$

*Proof.* If $p = 0$, or $p \geq m$, then $u(p) \geq 1 \geq \max(\frac{1}{m}, \frac{1}{n})$. Otherwise, $1 \leq p < m$, which is assumed for the rest of this proof.

Now $u(p) = \frac{p}{m} + \frac{1}{m}\binom{m}{p+1}n^{-p} \geq \frac{p}{m} \geq \frac{1}{m}$. If $pn \geq m$, then $u(p) \geq \frac{p}{m} \geq \frac{p}{pn} = \frac{1}{n}$. Otherwise $m = pn + q$ for some integer $q \geq 1$, which is assumed for the rest of the proof. So:

$$
\begin{aligned}
u(p) &= \frac{pn^p + \binom{m}{p+1}}{mn^p} \\
&= \frac{pn^p + \frac{m}{p+1}\left(\prod_{r=1}^{p} \frac{m-r}{p+1-r}\right)}{mn^p} \\
&\geq \frac{pn^p + \left(\prod_{r=1}^{p} \frac{m-r}{p+1-r}\right)}{mn^p} \\
&= \frac{pn^p + \left(\prod_{r=1}^{p-1} \frac{m-r}{p+1-r}\right)(m-p)}{mn^p} \\
&= \frac{pn^p + \left(\prod_{r=1}^{p-1} \frac{pn+q-r}{p+1-r}\right)(pn+q-p)}{mn^p} \\
&= \frac{pn^p + \left(\prod_{r=1}^{p-1} \left(n + \frac{q-r-n+rn}{p+1-r}\right)\right)(p(n-1)+q)}{mn^p} \\
&\geq \frac{pn^p + \left(\prod_{r=1}^{p-1} \left(n + \frac{q-r-n+rn}{p+1-r}\right)\right)q}{mn^p} \\
&= \frac{pn^p + q\prod_{r=1}^{p-1} \left(n + \frac{(q-1)+(r-1)(n-1)}{p+1-r}\right)}{mn^p} \\
&\geq \frac{pn^p + q\prod_{r=1}^{p-1} n}{mn^p} \\
&= \frac{pn(n^{p-1}) + qn^{p-1}}{mn^p} \\
&= \frac{(pn+q)n^{p-1}}{(pn+q)n^p} \\
&= \frac{1}{n}
\end{aligned}
\tag{12}
$$

□

# 4 Continuous upper bounds

In this section, the parameter $p$ in the discrete upper bounds are specialized in a manner dependent on the source domain size $|S| = m$ and remixed

range size $|R| = n$. The goal is to find further upper bounds upon $u(p)$ which are both usefully low and amenable to easy-to-verify proofs.

At a large scale, we choose $p$ not far past the typical peak preimage size for a random function $f : S \to R$. On the small scale, $p$ is chosen in order to obtain easily verifiable upper bounds on $u(p)$. If the $p$ happens to nearly minimize $u(p)$, well, that is just lucky side effect. Similarly, if the resulting upper bounds are close to the lower bounds, then the tightness is again just a lucky side effect.

### 4.1 Significantly expanded keys

Suppose that $|R|/|S| = n/m \gg 1$, which means that the remixed key space is much larger than than the source key space. This covers the case in which the key remixing procedure is significantly expanding. In this case, setting $p = 1$ yields a usefully low upper bound:

$$\mu \leq u(1) = \frac{1}{m} + \frac{1}{mn}\binom{m}{2} = \frac{1}{m} + \frac{m-1}{2n} \leq \frac{1}{m} + \frac{m}{2n}, \tag{13}$$

which proves the next theorem.

**Theorem 2.** *If $n \geq m^3$, then:*

$$\frac{1}{m} \leq \mu \leq \frac{1}{m}\left(1 + \frac{1}{2m}\right); \tag{14}$$

*and if $n \geq m^2$, then:*

$$\frac{1}{m} \leq \mu \leq \frac{3}{2}\frac{1}{m}. \tag{15}$$

In other words, if remixed key length is expanded at least three-fold over the source key length, and if $m$ is large, then the upper and lower bounds are close proportionally. If the remixed key length is at least double the source key length, and $m$ is larger, the upper bound means that the remixed key is not substantially easier to surmise than the source key. The purpose of mentioning these bounds is not their tightness, but rather the simplicity of their proofs.

### 4.2 Moderately expanded keys

The previous bound took $p = 1$ but presumed the significant expansion: $n \geq m^2$. Taking $p = 1$ results in somewhat useful for slightly lesser amounts of expansion, but as $n/m$ goes down to one, the bound $u(1)$ starts to approach

$\frac{1}{2}$, which is a useless bound. So, in the case of moderate expansion, $m < n \leq m^2$, we try using larger values of $p$ to find better upper bounds.

Our proofs now start using two other tools: the logarithm, whose basic properties such as order-preservation and $\log(ab) = \log(a) + \log(b)$ and $\log(a^b) = b\log(a)$, we take for granted; and rounding reals up to integers, as in $\lceil 1.3 \rceil = 2$.

**Theorem 3.** *If $n > m$, then*

$$\mu \leq \frac{1}{m}\left(2 + \frac{1}{\frac{\log n}{\log m} - 1}\right). \tag{16}$$

*Proof.* Let

$$p = \left\lceil \frac{1}{\frac{\log n}{\log m} - 1} \right\rceil, \tag{17}$$

which is a non-negative integer because $n > m$. Therefore:

$$\begin{aligned}
\mu &\le u(p) \\
&= \frac{p}{m} + \frac{1}{m}\binom{m}{p+1}n^{-p} \\
&= \frac{p}{m} + \frac{1}{m}\left(\frac{m(m-1)\dots(m-p)}{(p+1)!}\right)n^{-p} \\
&\le \frac{p}{m} + \frac{1}{m}\frac{m^{p+1}}{(p+1)!}n^{-p} \\
&= \frac{p}{m} + \frac{1}{m}\frac{1}{(p+1)!}m^{p+1-p\frac{\log n}{\log m}} \\
&= \frac{p}{m} + \frac{1}{m}\frac{1}{(p+1)!}m^{1-p\left(\frac{\log n}{\log m}-1\right)} \\
&= \frac{p}{m} + \frac{1}{m}\frac{1}{(p+1)!}m^{1-\left\lceil\frac{1}{\frac{\log n}{\log m}-1}\right\rceil\left(\frac{\log n}{\log m}-1\right)} \qquad (18) \\
&\le \frac{p}{m} + \frac{1}{m}\frac{1}{(p+1)!}m^{1-\frac{1}{\frac{\log n}{\log m}-1}\left(\frac{\log n}{\log m}-1\right)} \\
&= \frac{p}{m} + \frac{1}{m}\frac{1}{(p+1)!} \\
&\le \frac{p}{m} + \frac{1}{m} \\
&= \frac{1}{m}\left(\left\lceil\frac{1}{\frac{\log n}{\log m}-1}\right\rceil + 1\right) \\
&\le \frac{1}{m}\left(2 + \frac{1}{\frac{\log n}{\log m}-1}\right)
\end{aligned}$$

where the third inequality above follows from the fact that $-\lceil b\rceil \le -b$ for any real number $b$ and the fact that $\frac{\log n}{\log m} - 1 > 0$. □

For small values of $p$ in this bound, this choice of $p$ is related to a generalization of the birthday surprise effect.

As $n$ approaches $m$ in Theorem 3, the upper bound gets very loose, actually becoming useless. (For $n \approx m + \log m$, the upper bound is in the neighborhood of 1, which is not very useful, and for $n = m + 1$, the bound usually exceeds 1, which is useless for success rates.) Nevertheless, the bound can be fairly useful, in cryptography, if the key length expansion is just one bit, which is formalized as follows.

**Theorem 4.** *If $n \geq 2m$, then*

$$\mu \leq \frac{2 + \log_2(m)}{m}. \tag{19}$$

*Proof.* Starting from Theorem 3:

$$\begin{aligned}
\mu &\leq \frac{1}{m}\left(2 + \frac{1}{\frac{\log n}{\log m} - 1}\right) \\
&\leq \frac{1}{m}\left(2 + \frac{1}{\frac{\log 2m}{\log m} - 1}\right) \\
&= \frac{1}{m}\left(2 + \frac{1}{\frac{\log 2 + \log m}{\log m} - 1}\right) \\
&= \frac{1}{m}\left(2 + \frac{1}{\frac{\log 2}{\log m}}\right) \\
&= \frac{1}{m}\left(2 + \frac{\log m}{\log 2}\right) \\
&= \frac{1}{m}\left(2 + \log_2 m\right)
\end{aligned} \tag{20}$$

where the second line inequality follows from $n \geq 2m$, with the direction of the inequality reversed since it appears in a denominator. □

### 4.3 Significantly compressed keys

If $|S|/|R| = m/n \gg 1$, then the source key space is much larger than the remix key space. In other words, the remixing procedure does significant key compression.

The next old and well-known lower bound on factorials uses another proof tool: some basic calculus, namely differentiation and integration.

**Lemma 1** (Stirling?)**.** *If $k$ is a positive integer, then $k! \geq e(k/e)^k$.*

*Proof.* By direct calculation:

$$\begin{aligned}
k! &= \exp \log k! \\
&= \exp \log \prod_{j=2}^{k} j \\
&= \exp \sum_{j=2}^{k} \log j \\
&= \exp \int_{1}^{k} \log \lceil x \rceil dx \\
&\geq \exp \int_{1}^{k} \log(x) dx \\
&= \exp \int_{1}^{k} \left( \frac{d}{dx} \left( x \log(x) - x \right) \right) dx \\
&= \exp \left( (k \log(k) - k) - (1 \log(1) - 1) \right) \\
&= \exp \left( (k \log(k) - k) + 1 \right) \\
&= e k^k / e^k
\end{aligned} \tag{21}$$

□

This lower bound on factorials is useful for providing an upper bound on the inverse of factorials, which in turn provides an upper bound on binomial coefficients with larger arguments. Now put $p = q(m/n) - 1$, where $q \geq e$. Then:

$$\begin{aligned}
u(p) &= \frac{p}{m} + \frac{1}{m} \frac{m(m-1)\dots(m-p)}{(p+1)!} n^{-p} \\
&\leq \frac{q}{n} - \frac{1}{m} + \frac{1}{m} \frac{m^{p+1}}{(p+1)!} n^{-p} \\
&= \frac{q}{n} - \frac{1}{m} + \frac{n}{m} \frac{1}{(p+1)!} \left( \frac{m}{n} \right)^{p+1} \\
&\leq \frac{q}{n} - \frac{1}{m} + \frac{n}{m} \frac{1}{e((p+1)/e)^{p+1}} \left( \frac{m}{n} \right)^{p+1} \\
&= \frac{q}{n} - \frac{1}{m} + \frac{n}{em} \left( \frac{(m/n)}{q(m/n)/e} \right)^{p+1} \\
&= \frac{q}{n} - \frac{1}{m} + \frac{n}{em} \left( \frac{e}{q} \right)^{p+1} \\
&\leq \frac{q}{n} - \frac{1}{m} + \frac{n}{em},
\end{aligned} \tag{22}$$

where the last inequality follows from $q \geq e$. If $m \geq n^2$, then we get:

$$\mu \leq u(p) \leq (q + 1/e)\frac{1}{n} \tag{23}$$

The smallest we can make $q$ while satisfying $q \geq e$ is $q = \lceil \frac{em}{n} \rceil \frac{n}{m} \leq e + \frac{n}{m} \leq e + 1/n$. This proves the following theorem:

**Theorem 5.** *If $n \leq \sqrt{m}$, then*

$$\frac{1}{n} \leq \mu \leq \frac{e + \frac{1}{e} + \frac{1}{n}}{n}. \tag{24}$$

### 4.4 Easier and looser bounds for compressed keys

If one wants to avoid basic calculus but one is willing to consider square roots, then the following (well-known: [GKP94]) weak lower bound on factorials can be used:

**Lemma 2.** *If $k$ is a non-negative integer, then $k! \geq (\sqrt{k})^k$.*

*Proof.* Calculating directly:

$$\begin{aligned}
k! &= \prod_{j=1}^{k} j \\
&= \prod_{j=1}^{k} \sqrt{j}\sqrt{j} \\
&= \left(\prod_{j=1}^{k} \sqrt{j}\right)\left(\prod_{j=1}^{k} \sqrt{k+1-j}\right) \\
&= \prod_{j=1}^{k} \sqrt{j(k+1-j)} \\
&= \prod_{j=1}^{k} \sqrt{k + (j-1)(k-j)} \\
&\geq \prod_{j=1}^{k} \sqrt{k} \\
&= \sqrt{k}^k
\end{aligned} \tag{25}$$

□

Using this we get the occasionally useful bound:

**Theorem 6.** *For all $m, n \geq 1$:*

$$\mu \leq \frac{m}{n^2} + \frac{n}{m}. \tag{26}$$

*Proof.* Let:

$$p = \left\lceil \left(\frac{m}{n}\right)^2 \right\rceil - 1, \tag{27}$$

which is a non-negative integer. Hence:

$$\begin{aligned}
\mu &\leq u(p) \\
&= \frac{p}{m} + \frac{1}{m}\binom{m}{p+1} n^{-p} \\
&\leq \frac{p}{m} + \frac{1}{m}\frac{m^{p+1}}{(p+1)!} n^{-p} \\
&= \frac{p}{m} + \frac{n}{m}\frac{1}{(p+1)!}\left(\frac{m}{n}\right)^{p+1} \\
&\leq \frac{p}{m} + \frac{n}{m}\frac{1}{\sqrt{p+1}^{p+1}}\left(\frac{m}{n}\right)^{p+1} \\
&= \frac{p}{m} + \frac{n}{m}\left(\frac{m/n}{\sqrt{p+1}}\right)^{p+1} \\
&= \frac{p}{m} + \frac{n}{m}\left(\frac{m/n}{\sqrt{\lceil (m/n)^2 \rceil}}\right)^{p+1} \\
&\leq \frac{p}{m} + \frac{n}{m}\left(\frac{m/n}{\sqrt{(m/n)^2}}\right)^{p+1} \\
&= \frac{p}{m} + \frac{n}{m} \\
&\leq \frac{(m/n)^2}{m} + \frac{n}{m} \\
&= \frac{m}{n^2} + \frac{n}{m}.
\end{aligned} \tag{28}$$

□

This bound is generally quite loose compared to other bounds, even where it does provide useful bounds, but its main defect for us is that the set of pairs $(m, n)$ for which it provides useful is rather limited.

### 4.5 Moderately compressed keys

In proving many of the previous bounds, a term of the form $n/m$ or $m/n$ appeared. As $n/m$ approaches one, this term will cause a bound to become to close to one, which is uselessly large. So, a better choice of $p$ is needed, and the following lemma will help, which is the last proof tool needed.

**Lemma 3** (Historical?)**.** *Let $x \geq e^e$. If*

$$y \geq \frac{\log x}{\log\log x - \log\log\log x}, \tag{29}$$

*then $y^y \geq x$.*

*Proof.* Calculating:

$$\begin{aligned} y^y &= \exp(y \log y) \\ &\geq \exp\left(\frac{\log x}{\log\log x - \log\log\log x}\left(\log\log x - \log\left(\log\log x - \log\log\log x\right)\right)\right) \\ &\geq \exp\left(\frac{\log x}{\log\log x - \log\log\log x}\left(\log\log x - \log\left(\log\log x\right)\right)\right) \\ &= \exp(\log x) \\ &= x, \end{aligned} \tag{30}$$

where the second inequality is derived as follows. The given condition $x \geq e^e$ gives $\log\log\log(x) \geq 0$, which gives $\log\log x - \log\log\log x \leq \log\log x$. This gives $-\log(\log\log x - \log\log\log x) \geq -\log(\log\log x)$. □

The right hand side of inequality (29) should not be seen as something fundamentally important, but rather just a bound for something rather mundane described using the familiar logarithm functions. We could instead use the less familiar, but still notionally mundane, inverse of the function $x \mapsto x^x$. This inverse can be bounded numerically quite easily using a binary search or Newton's method. The expression involving repeated logarithms is just slightly more convenient to write down and to evaluate.

**Theorem 7.** *If $e^e \leq n \leq em$, then*

$$\mu \leq \frac{1}{n}\left(1 + \frac{e \log n}{\log\log n - \log\log\log n}\right). \tag{31}$$

*Proof.* Let $c = em/n$. Let

$$p = \left\lceil \frac{c \log n}{\log\log n - \log\log\log n} \right\rceil - 1. \tag{32}$$

Let $y = (p+1)/c$, then

$$\begin{aligned} y &= \frac{1}{c}\left\lceil \frac{c \log n}{\log\log n - \log\log\log n} \right\rceil \\ &\geq \frac{1}{c}\left( \frac{c \log n}{\log\log n - \log\log\log n} \right) \\ &= \frac{\log n}{\log\log n - \log\log\log n}, \end{aligned} \tag{33}$$

which implies that $y^y \geq n$, by applying the previous lemma. Since $p$ is a non-negative integer, we get:

$$\begin{aligned} \mu &\leq u(p) \\ &= \frac{p}{m} + \frac{n}{m}\binom{m}{p+1}\frac{1}{n^{p+1}} \\ &\leq \frac{p}{m} + \frac{n}{m}\frac{m^{p+1}}{(p+1)!}\frac{1}{n^{p+1}} \\ &\leq \frac{p}{m} + \frac{n}{m}\frac{(m/n)^{p+1}}{e((p+1)/e)^{p+1}} \\ &= \frac{p}{m} + \frac{n}{em}\left(\frac{(em/n)}{p+1}\right)^{p+1} \\ &= \frac{p}{m} + \frac{1}{c}\left(\frac{c}{cy}\right)^{cy} \\ &= \frac{p}{m} + \frac{1}{c}\left(\frac{1}{y^y}\right)^{c} \\ &\leq \frac{p}{m} + \frac{1}{c}\left(\frac{1}{n}\right)^{c} \\ &= \frac{1}{m}\left(\left\lceil \frac{c \log n}{\log\log n - \log\log\log n} \right\rceil - 1\right) + \frac{1}{c}\left(\frac{1}{n}\right)^{c} \\ &\leq \frac{1}{m}\left( \frac{c \log n}{\log\log n - \log\log\log n} \right) + \frac{1}{c}\left(\frac{1}{n}\right)^{c} \\ &= \frac{1}{n}\left( \frac{e \log n}{\log\log n - \log\log\log n} \right) + \frac{1}{c}\left(\frac{1}{n}\right)^{c} \\ &\leq \frac{1}{n}\left( \frac{e \log n}{\log\log n - \log\log\log n} \right) + \frac{1}{n}, \end{aligned} \tag{34}$$

where the last inequality above follows from $c \geq 1$. $\square$

# 5 Numerical upper bounds

In this section, we use the previous results to deduce some numerical bounds.

## 5.1 Significant expansion

For a concrete example, suppose that the source key has a length of 128 bits, and remixed key has length expanded to 256 bits. Upper bound (15) says an adversary Eve, even with unlimited computational power, has probability at most $2^{-127.4}$ of surmising a remixed key (in one try).

To keep all the claims in this report provable purely by pencil-and-paper, start from (15) of Theorem 2, and work as follows:

$$\begin{aligned}
\mu &\leq \frac{3}{2}\frac{1}{m} \\
&= \frac{3}{2}\frac{1}{2^{128}} \\
&= (3)2^{-129} \\
&= (3^5)^{(1/5)}2^{-129} \\
&= (243)^{(1/5)}2^{-129} \\
&\leq (256)^{(1/5)}2^{-129} \\
&= (2^8)^{(1/5)}2^{-129} \\
&= 2^{8/5-129} \\
&= 2^{1.6-129} \\
&= 2^{-127.4}
\end{aligned} \tag{35}$$

For an easier but looser numeric bound: $\mu \leq \frac{3}{2}\frac{1}{m} \leq 2\left(\frac{1}{m}\right) = 2^{1-128} = 2^{-127}$. Of course, the actual probability could be as low as $2^{-128}$, at least according the lower bounds proven so far in this report.

Obviously, a key chosen uniformly at random from the expanded key space is much harder to surmise ($2^{-256}$) than a remixed key in the expanded key space ($2^{-128}$). On one hand, strictly speaking, one ought not exaggerate by saying that remixed expanded key is as strong as any key of that size: rather, just say that it is strong enough. On the other hand, this report has not offered any reasons why such an exaggeration may cause harm.

### 5.2 Moderate expansion

For a first numerical example, suppose that $|S| = m = 2^{128}$ and $|R| = n = 2^{192}$. Starting from Theorem 3:

$$\begin{aligned}
\mu &\leq \frac{1}{2^{128}} \left( 2 + \frac{1}{\frac{\log 2^{192}}{\log 2^{128}} - 1} \right) \\
&= 2^{-128} \left( 2 + \frac{1}{\frac{192}{128} - 1} \right) \\
&= 2^{-128} \left( 2 + \frac{1}{\frac{3}{2} - 1} \right) \\
&= 2^{-128} \left( 2 + 2 \right) \\
&= 2^{-128} 2^2 \\
&= 2^{-126}.
\end{aligned} \tag{36}$$

For a second numerical example, put $m = 2^{128}$ and $n \geq 2^{129}$. Then Theorem 4 implies that:

$$\mu \leq 2^{-128}(2 + \log_2(2^{128})) = 2^{-128}(130) \leq 2^{-128} 2^8 = 2^{-120}. \tag{37}$$

### 5.3 Significant compression

Putting $m = 2^{256}$ and $n = 2^{128}$ in Theorem 5 gives $\mu \leq 2^{-126.37}$, if one permits some routine calculations involving $e$. But, again, to keep all the report's results verifiable by pencil-and-paper, we prove a slightly weaker numerical bound by hand. We must use calculus again, because the constant $e$ appearing in the theorem is defined in terms of calculus:

$$e^x = \sum_{j \geq 0} x^j / j! \tag{38}$$

and therefore:

$$\begin{aligned} e + 1/e &= e^1 + e^{-1} \\ &= \sum_{j \geq 0} \frac{1^j + (-1)^j}{j!} \\ &= \sum_{j \geq 0} \frac{2}{(2j)!} \\ &\leq \frac{2}{0!} + \frac{2}{2!} + \sum_{j \geq 2} \frac{2}{4!4^{j-1}} \\ &= 2 + 1 + \frac{2}{24} \left(1 - \frac{1}{4}\right)^{-1} \\ &= 3 + 1/9 \end{aligned} \tag{39}$$

Then $e + 1/e + 1/n \leq 3 + 1/8 = 25/8$ for large enough $n$. So,

$$\begin{aligned} \mu &\leq \frac{25}{8} \frac{1}{n} \\ &= \frac{5^2}{2^3} \frac{1}{2^{128}} \\ &= (5^2) 2^{-131} \\ &= (5^6)^{(1/3)} 2^{-131} \\ &= (15625)^{(1/3)} 2^{-131} \\ &\leq (16384)^{(1/3)} 2^{-131} \\ &= (2^{14})^{(1/3)} 2^{-131} \\ &= 2^{14/3 - 131} \\ &= 2^{4.666\cdots - 131} \\ &= 2^{-126.333\ldots}. \end{aligned} \tag{40}$$

Such a small probability is, of course, more than adequate for security against a surmisal attack. A tighter bound is probably possible in this case, because this report has aimed for simpler proofs rather than tighter bounds. A tighter bound does not result in a substantial gain for cryptographic applications.

### 5.4 A loose numeric bound

Let $(m, n) = (2^{192}, 2^{128})$. Theorem 6 gives:

$$\begin{aligned}\mu &\leq \frac{m}{n^2} + \frac{n}{m} \\ &= \frac{2^{192}}{2^{256}} + \frac{2^{128}}{2^{192}} \\ &= 2^{-64} + 2^{-64} \\ &= 2^{-63}.\end{aligned} \tag{41}$$

This is much looser than the next numeric bound, but recall that Theorem 6 was provided because it avoided basic calculus, and thus has an arguably simpler proof.

### 5.5 Moderate compression

Suppose that $n = 2^{128}$, and $m = 2^{128}$ or $m = 2^{192}$, both of which have $e^e \leq n \leq em$. Then, using Theorem 7 and the bound $\log 2 \geq \log\log 2$, we get:

$$\begin{aligned}\mu &\leq \frac{1}{2^{128}}\left(1 + \frac{e\log 2^{128}}{\log\log 2^{128} - \log\log\log 2^{128}}\right) \\ &\leq \frac{1}{2^{128}}\left(1 + \frac{e128\log 2}{\log(128\log 2) - \log\log(128\log 2)}\right) \\ &= \frac{1}{2^{128}}\left(1 + \frac{e128\log 2}{7\log 2 + \log\log 2 - \log(7\log 2 + \log\log 2)}\right) \\ &\leq \frac{1}{2^{128}}\left(1 + \frac{e128\log 2}{7\log 2 + \log\log 2 - \log(8\log 2)}\right) \\ &= \frac{1}{2^{128}}\left(1 + \frac{e128\log 2}{7\log 2 + \log\log 2 - 3\log 2 - \log\log 2}\right) \\ &= \frac{1}{2^{128}}\left(1 + \frac{e128\log 2}{4\log 2}\right) \\ &= 2^{-128}\left(1 + 32e\right) \\ &\leq 2^{-128}\left(1 + 32(3)\right) \\ &= 2^{-128}\left(97\right) \\ &\leq 2^{-128}(2^7) \\ &= 2^{-121},\end{aligned} \tag{42}$$

using the bound $e \leq 3$ near the end.

### 5.6 A general numerical result

Common cryptographic practice aims for keys of length at least 128 bits. So, we formulate the relevant numerical result:

**Theorem 8.** *If* $|R|, |S| \geq 2^{128}$*, then*

$$\mu \leq 2^{-120}. \tag{43}$$

*Proof.* One of $n/m \geq 2$ or $n/m \leq e$ is true because $2 < e$. If $n/m \geq 2$, then, Theorem 4 says:

$$\mu \leq \frac{2 + \log_2 m}{m}. \tag{44}$$

If $n/m \leq e$, then, Theorem 7 says:

$$\mu \leq \frac{1}{n}\left(1 + \frac{e \log n}{\log\log n - \log\log\log n}\right). \tag{45}$$

These bounds can be seen to be decreasing functions[1] of $m$ and $n$ respectively, at least when $m, n \geq 2^{128}$, so both are at most their values at $m = n = 2^{128}$. We saw earlier in (37) that the first takes value at most $2^{-120}$ at $m = 2^{128}$ and, in (42), that the second is at most $2^{-121} \leq 2^{-120}$ at $n = 2^{128}$. So, $\mu \leq 2^{-120}$ in all cases. □

A much larger upper bound, such as $2^{-50}$, would likely suffice for cryptographic security.[2][3] The fact that a better bound was achieved is more or less a free by-product of the proof techniques: not by some extra effort on our part. More precisely, in deriving this result, this was the first rigorous (and first correct numeric) upper bound that I attained.

Lowering this upper bound would be a good academic exercise, and might have applications outside cryptography. Conversely, simplifying the proofs of this report would be another good academic exercise, and might have educational applications. Perhaps differentiation and integration can be avoided altogether.

Fortunately, probabilities cannot erode over time, unlike the cost of computation. In other words, they are almost absolute. If, for some reason, one needs to use larger keys, to resist faster computers or better attack algorithms, then one is probably not obliged to update the result above.

[1] This step is bigger than any of the steps of the previous proofs.

[2] I leave analysis of the situation in the multi-user setting to other work.

[3] I leave the notion of computing the expected losses of surmised keys to other work.

## 6 Conclusion

A proof-demanding pessimist need never again worry that key derivation somehow gives an adversary a decent chance, such as $2^{-30}$, to the extent the remix models such derived keys. Naturally, the pessimist will still doubt whether any deterministic pseudorandom function, even with uniformly random salt and random secret source key, is well-modeled by remixed keys. Hopefully other research has addressed or will address that concern.

A proof-appreciating optimist should find the numeric unsurmisability pleasantly unsurprising.

## References


[NIST 800-90C] E. Barker and J. Kelsey. *Recommendation for Random Bit Generator (RBG) Constructions*, Special Publication 800-90. National Institute of Standards and Technology, Aug. 2012.

[DPW13] Y. Dodis, K. Pietrzak and D. Wichs. *Key derivation without entropy waste.* ePrint 2013/338, International Association for Cryptologic Research, Oct. 2013. http://eprint.iacr.org/2013/338.

[FS09] P. Flajolet and R. Sedgewick. *Analytic Combinatorics.* Cambridge University Press, 2009.

[GKP94] R. L. Graham, D. E. Knuth and O. Patashnik. *Concrete Mathematics: A Foundation for Computer Science.* Addison–Wesley, 2nd edn., 1994.

[Kra10a] H. Krawczyk. *Cryptographic extraction and key derivation: The HKDF scheme.* ePrint 2010/264, International Association for Cryptologic Research, May 2010. Full version of [Kra10b].

[Kra10b] ———. *Cryptographic extraction and key derivation: The HKDF scheme.* In T. Rabin (ed.), *Advances in Cryptology — CRYPTO 2010*, Lecture Notes in Computer Science 6223, pp. 631–648. International Association for Cryptologic Research, Springer, Aug. 2010. Extended abstract of [Kra10a].

[KSC78] V. F. Kolchin, B. A. Sevast'yanov and V. P. Chistyakov. *Random Allocations*. Scripta Series in Mathematics. V. H. Winston & Sons, Halsted Press, 1978. Translation by A. V. Balakrishnan.

[Sho09] V. Shoup. *A Computational Introduction to Number Theory and Algebra*. Cambridge University Press, 2nd edn., 2009.

# A Motivation

## A.1 Salt and hash: specific and relative surmisability

A random function is not a realistic object for large sets $R$ and $S$, since describing it requires too much information to specify, to determine and to communicate. So, in practice, something closer to the following is used.

Let $T$ be a finite non-empty set. Let $d \in (R^S)^T$ be a fixed function from $T$ to $R^S$. In $d$-remixing, Alice and Bob have a shared secret $s \in_\$ S$. They determine a one-time value $t \in_\$ T$ called the *salt*. The salt must be unpredictable to an adversary, and if the two parties are establishing $r$, then the salt must be established in some authenticated manner: the adversary must not be able manipulate the salt, as this could allow the adversary to render it more predictable. They compute the $d$-remixed key $r = d(t)(s)$, by applying the function $d(t)$ to $s$. It is not necessary to represent the actual function $d(t)$. Rather, it suffices to be able to evaluate of $d(t)(s)$, which we may written as $d(t, s)$, when clear from context, to emphasize this fact.

If we put $T = R^S$ and let $d$ be the identity function, then we recover our original formalism for remixed keys, because $d(t, s) = d(t)(s) = t(s)$, and just re-name $t$ to $f$.

The *$d$-specific surmisability* is defined to be:

$$\mu(d) = \max_{e \in R^{R^S}} \Pr\left[(s, t) \in_\$ S \times T : e(d(t)) = d(t)(s)\right]. \tag{46}$$

This definition is formulated to align with the original definition, but one can simplify it as follows:

$$\mu(d) = \max_{e' \in R^T} \Pr\left[(s, t) \in_\$ S \times T : e'(t) = d(t, s)\right], \tag{47}$$

by defining $e' = e \circ d$.

We can also formulate the $t$-conditional $d$-specific surmisability $\mu(d)|_t$ as Eve's success rate after she sees $t$. This quantity is closely related to

the previously discussed $f$-conditional surmisability: $\mu(d)|_t = \mu|_{d(t)}$. Also if $|T| = 1$, or if $d$ is a constant function (not varying with its input $t$), then $d$-specific surmisability is $d(t)$-conditional surmisability: $\mu(d) = \mu|_{d(t)}$.

The *relative surmisability* of $d$ is defined to be $\rho(d) = \frac{\mu(d)}{\mu}$. If the relative surmisability is one, then $d$ provides the same resistance as random function against surmisal attacks on the remixed key. It is possible that $\rho(d) < 1$, for example, if $d(t)$ is always a permutation. In practice, the most efficient permutations $d(t)$ are often efficiently invertible given $t$, which may sometimes have negative security effects not related to surmisability. So generally, one expects that $\rho(d) \leq 1$, at least approximately. If $\rho(d) > 1$, then the $d$-remixed keys are easier to surmise then uniformly remixed keys, and are thus are weaker to some extent.

The relative surmisability is an information-theoretic property of $d$. It does not depend on any computational hardness assumptions about $d$, but rather the shape of $d$ as a function, where shape means the isomorphism class of the functional digraph of $d$.

For a concrete example, let $T = S$ be the set of bit strings of length 128. Let $h$ be a standard hash function, specifically either SHA1 or SHA256. Let $d_{h,*}(t, s)$ be the first 128 bits of $h(t\|s)$, where $t\|s$ is the concatenation of $t$ and $s$. Consider the following concrete conjecture:

**Conjecture 1.** *The relative surmisability of $d_{h,*}$ satisfies $\rho(d_{h,*}) \leq 2^{70}$.*

To review, this conjecture is saying that $d_{h,*}(t, \cdot)$ has an expected maximum preimage size at $2^{70}$ times larger than average for a random function. This seems like a modest conjecture, though I have no idea how to prove it.

This conjecture implies that $\mu(d_{h,*}) \leq 2^{70}\mu$. Combined with Theorem 8, this says that $\mu(d_{h,*}) \leq 2^{-50}$, which may be adequate for security of $d_{h,*}$-remixed keys, at least against surmisal attacks.

If we de-randomize further and set $|T| = 1$, so $T$ consists of bit strings of length zero, making $d_{h,-}(t, s)$ the first 128 bits of $h(s)$, then we can make a similar conjecture:

**Conjecture 2.** *The relative surmisability of $d_{h,-}$ satisfies $\rho(d_{h,-}) \leq 2^{70}$.*

If this conjecture fails, then we can find a collision in the 128-bit prefix of the hash function $h$ by trying about $2^{51}$ random values of $s$. In other words, this conjecture is plausible under nearly standard conjectures about hash functions SHA1 and SHA256.

## A.2 Why remix keys?

Given that a remixed key seems to be slightly easier to surmise than the source key or a uniform key in the remix space, at least to the extent that we can upper bound, a natural question is: why bother remixing at all?

Some protocols, like the TLS protocol, already do something like remixing, so the question is perhaps best asked of the TLS working group. In other words, the question could be deemed somewhat out of scope of this report. Nevertheless, I now suggest what I think may the kind of reasons protocols have for remixing:

- The source key space $S$, when encoded into $R$, including and padding or truncation, with a non-random function, results in very biased keys in $R$. Such bias may be exploitable for some applications of the key $r$.
- In some systems, a source key may be a precious resource that one would not like to use directly in a bulk cryptographic algorithm (because of the risk of side channels).
- In some protocols, the remixing procedure may provide some (extra) protection against replay attacks.
- In some systems, exposure of the source key may incur a security risk, and remixing (or rehashing) helps in the sense of applying a one-way function.

So, in many cases, the remixing effect is side effect of some other cryptographic function. Understanding the impact of remixing may help more formally understand the trade-offs involved.

## A.3 Truth in advertising

It may be more than fair to say that the 128-bit AES keys modeled by remixed keys provide 128 bits of security, even though the remixed AES keys are not strictly uniform distribution. This report helps support this interpretation to an extent: it provides a numerical upper bound on the unsurmisability of a remixed key, at least when both the source key and the remixed key are at least 128 bits in length.

## A.4 A question of importance

An important question is whether the question of surmisability is an important question. The importance depends on the answer. Essentially, if

surmisability happens to be a large value like $2^{-20}$, then surmisability would be a somewhat important question, because it would show that one might need to rely on a extra computational assumption to get adequate security. Fortunately, the answer to the surmisability is much better than that. So, by the answering the surmisability question positively, we have rendered the question unimportant, in the sense that we had no reason to worry about it having a negative answer.

# B More variants on surmisability

## B.1 Multiple guesses and key searching

In many situations, an adversary Eve gets the opportunity to see Alice or Bob use $r$ in such a way that Eve can test any value $r' \in R$ for whether $r' = r$. Usually, she can do these tests off-line, but in some unusual protocols Eve might only be able to make these test using some interaction with Alice or Bob (especially in contexts where the keys are treated as passwords).

Generally, the way Eve does this is as follows: for each guessed key, Eve tries to implement the next step in the protocol that uses $r$. For example, the protocol may use to verify a message authentication code (MAC) tag. Or, it may use $r$ to decrypt a ciphertext, in which case Eve inspect the plaintext decryption to see whether it seems to have the proper formatting expect of the real plaintext. Or, the next step may be respond to some challenge, interactively proving knowledge of $r$.

To formalize these ideas, first let $g$ be a non-negative integer, and consider the *g-surmisability* (or *g-searchability*):

$$\mu_g = \max_{e \in (R^G)^{R^S}} \Pr\left[(s, f) \in_\$ S \times R^S : f(s) \in e(f)(G)\right], \tag{48}$$

where $G$ is any fixed set of size $g$, and $e(f)(G)$ is the image of $G$ under the function $e(f)$ . The set $e(f)(G)$ represents Eve's set of guesses for $r$. Then, surmisability is just 1-surmisability (1-searchability) under this definition: $\mu_1 = \mu$.

A more computational way to measure Eve's success is to account for the effort she needs to confirm each guess, which adds to a lot of effort when the number of guesses is high. The *aegis* (of remixing $S$ to $R$) is

$$\eta = \max_{e \in (R^R)^{R^S}} E\left[(s, f) \in_\$ S \times R^S : i(e(f)(f(s)), R)\right], \tag{49}$$

where $i(u, V)$ being the index of element $u$ in set $V$, under some fixed ordering $V$. Given $f$, Eve computes $e(f) \in R^R$. The function $e(f)$ re-orders

the elements in order of likelihood of being the remixed key. This number is the average number of guesses $r'$ at $r$ Eve needs to check in the theoretical remixing game. The higher the aegis, the more secure is the remixed key.

The aegis corresponds to exhaustive key search attacks, except that the adversary being modeled here has unlimited computational power in choosing the best strategy for the search based on seeing the remix function. Arguably, a large aegis, such as $2^{127}$ for a 128-bit key, corresponds more accurately to what cryptographers think of as 128-bit security. By contrast, a surmisability as large as $2^{-50}$ is hardly a concern, because: firstly, nobody expected the surmisability to be that low (a view which this report tries to rigorously support), and secondly, surmisability seems to be unaffected by the computational resources of an adversary. In fact, if one worries that an adversary's computational power will increase drastically, then one should seek a higher aegis.

I have not checked this, but I think $g$-surmisability and aegis are related as follows: $\eta = n - (\mu_1 + \mu_2 + \cdots + \mu_n)$ where $n = |R|$.

Though there are many good reasons for focusing on the aegis of remixed keys, that does not mean one should ignore surmisability. For example, in $d$-specific remixing, the aegis can be acceptably high, but the surmisability can be unacceptably high. For example, the remix function $d(t)$ might induce a spike of size $2^{-20}$ in the probability distribution of the keys, yet might induce completely a flat distribution beyond this single spike. This discrepancy may be due to over-weighing the higher index values of rarer guesses.

Perhaps this deficiency of aegis can be corrected by re-weighting the indices. For example, if one takes the expected value of the logarithm of the index, then this loosely models Moore's law, accounting for exponential increase in computational power over time. We may call this the *longevity* $\lambda$. Of course, it is easy to define such things, but it remains to be seen whether they are useful. I have have not verified it, but my rough estimate for the longevity would be $\lambda \approx \log(n) - (\mu_1/1 + \mu_2/2 + \cdots + \mu_n/n)$, and maybe the latter would serve as a more natural definition of longevity.

Bounding $\mu_g$ might be more difficult than bounding $\mu$. In fact, if I recall correctly, Kolchin *et al* [KSC78] provide some (asymptotic) balls-in-bins bounds related to the quantities $\mu_g$. Regardless, I think that there is much less reason for cryptographers to worry about a high value of $\mu_g$ for $g > 1$, for the following heuristic reasons.

Although remixing does indeed introduce a spike in the probability distribution of the remixed key, intuition expects the height of this spike to reduce as one gets away from the most likely value of remixed key. For

example, we might expect that $\mu_g/(g/n)$ would be smaller than $\mu_1/(1/n)$. In this report, we showed that $\mu$ is not dangerously higher than $1/n$. So, we have no reason to expect that $\mu_g$ is dangerously higher than the corresponding probability for the uniform source keys or uniform remix-range keys, and we expect $\mu_g$ to actually be relative closer to one those baseline probabilities $g/n$. In other words, remixing offers even less advantage on multiple-guesser (key-searcher) than it does to a single-guesser. Therefore, bounding $\mu_1$ seems like a more urgent problem than bounding $\mu_g$.

### B.2 Non-uniform source secrets

If the source secret $s$ is chosen from the set $S$ random but non-uniformly, then one can treat $s$ as a random variable. A variant of $\mu$ can still be defined, but will depend on the choice of random variable for $s$.

This variant can be used to model to things such as: user-chosen passwords; secret messages to be encrypted, from which one wants to derive some encryption keys; physical noise sources from which one would want to derive keys.

I am not sure how to proceed in bounding such a variable surmisability, but it may be a good exercise. Perhaps this has already be done, as in Dodis *et al* [DPW13].

### B.3 Computationally strong remixers and the quasimode problem

It is of course realistic to impose some computational limitations on the adversary Eve. For example, a real-world Eve may have difficulty in determining—and computing—the optimal function $e \in R^{R^S}$. An alternative approach would be to formulate a computational problem related to Eve's task here. An example is the *quasimode* problem: given $f$, find some output values that are significantly more likely than average. It seems plausible that, for random, or well-designed pseudorandom, function $f$, the quasimode problem is computationally infeasible. Furthermore, in cases were $f$ is actually a random function, then it would seem that the Eve would need at least a large number of calls to $f$ to solve the quasimode. This would essentially be random oracle setting. This report does not examine the difficulty of the quasimode problem any further.

## B.4 Distinguishing remixed keys from uniform keys

A rather different challenge for the adversary Eve is to distinguish the remixed key $r$ from a key chosen uniformly at random from $R$. This report does not discuss indistinguishability formally. I expect that the issue of distinguishability has already been well-resolved in existing publications on cryptography.

Intuitively, if the source key-space $S$ is much smaller than $R$, then an Eve with unlimited computational power can easily distinguish $r$ from a uniformly random key. In other remixed keys are non information-theoretically distinguishable from the uniformly random keys. Of course, they may well be indistinguishable to computationally-limited adversary. Basically, this seems to suggest that to expand a key, one needs a one-way function.

Conversely, $|S| = m$ is much larger than $|R| = n$, then it seems the statistical distance between the remixed key and a uniformly random range key is $O(\sqrt{n/m})$. Basically, this seems to suggest that to compress key, one may not need a one-way function.

The indistinguishability of the remixed is clearly important if the remixed key is used as a key stream, which often effectively exposes the remixed key. In many cases, however, the remixed key is used in algorithms that do not directly expose the key, such as symmetric encryption, or message authentication codes. In those applications, the need for indistinguishability is less dire.

## B.5 Iterated remixing

An simple question would be what happens if a single secret key $s$ is remixed into a multiple keys $r$. One can treat each individually, one combine them all into one key. If adversary is given any one of the remixed keys, then it becomes an altogether different problem, more like the distinguishing problem of the previous section.

A more unusual question would what happens if a remixed key is remixed itself, and so on, perhaps through multiple remixing generations. This becomes similar to the theory of iterated random functions, which is quite well-studied. Exact, non-asymptotic bounds might be interesting.

## B.6 Entropy and terminology

By considering logarithms of probabilities in various ways, one enters into the realm of entropy. Entropy is a very useful way to measure information. There are many kinds of entropy that can be defined for a single probability

distribution (or even a pair of random variables). I wanted this report to be self-contained and simple, so I did not describe the results in terms of entropy.

Loosely speaking, the negative logarithm of surmisability corresponds to what some people the conditional min-entropy of the even of the adversary surmising the remixed key condition on the random variable of the remix function. This is a slightly confusing terminology: because it does not align well with conditional surmisability (which is a conditional probability). (In my own, I used the term contingent entropy for this kind of thing. The concept of contingent working entropy corresponds to negative logarithm of $g$-surmisability.)

The probability that I have called surmisability might also be called guessability or predictability. I avoided terms derived from the verb guess because it does not emphasize the fact that we are considering the case where Eve has only a single guess. I avoided terms derived from predict, because the remix keys are derived from an unpredictable salt. The unpredictability of the salt is defined from some time before the remix function is made public to the adversary. But from this point, the remixed is unpredictable. The conditional surmisability is defined from a point in time after the remix function is made public. The surmisability can be viewed the average conditional surmisability, so it is not defined in terms of any point in time. So, to avoid confusion, I have used the rarer near-synonym for the verb guess, the verb surmise. (Some other alternatives are divine, deduce, ascertain, and so on, but these seem farther away in meaning.)

# C Previous work

In a draft standard, Barker and Kelsey [NIST 800-90C] make some non-asymptotic claims about deriving new keys from by applying a hash function. This report was, in part, inspired by those claims, though it diverged towards addressing a different problem.

Given the similarity of remixing to common cryptographic practices, one might infer an implicit folklore knowledge that remixed keys are not substantially easier to guess than one of the source key or random key.

It should be also be expected that the results of this report are explicitly or implicitly established in cryptographic publications. For example, the remixing model is quite similar to the random oracle model, except that the remixing allows an arbitrary amount of adversarial computation. In other words, it may well be the case that some proofs in the random oracle that

limit adversaries to a cost $t$ and probability $\epsilon$ are such that if $t$ is allowed to go to infinity, the $\epsilon$ approach a value that implies the upper bounds in this report (but perhaps not, if the larger value $t$ enables the adversary other more successful attacks).

Work on key derivation may be more directly applicable to the report. For example, Krawczyk [Kra10a] and Dodis, Pietrzak and Wichs, [DPW13] discuss entropy and key derivation. Unfortunately, I have only read a small sample of such work, and I failed to find the results of the nature described in this report.

Mathematically, it can be seen that $\mu$ depends on a very old mathematical problem, sometimes called the "balls-in-bins" problem.

So far, I have found three treatments of this problem.

- Kolchin, Sevast'yanov and Chistyakov's book *Random Allocations* [KSC78] solves this problem and some interesting variations of this problem (that might be related to what I called $g$-surmisability). I read this a few years ago, but no longer have easy access to the book. I did not fully understand their math, but I recall their results being asymptotic. Reading their work likely informed me in generalizing Shoup's results, though I did not recall or refer their results while writing this paper.

- Flajolet and Sedgewick [FS09, Proposition VIII.10] wrote a book on analytic combinatorics and include something about this result. I recall their proof methods to be far more sophisticated than those used in this report, though perhaps superior because of their wider applicability. I also recall their results to be asymptotic.

- Shoup's book [Sho09, Theorem 8.29] provides a one-page, self-contained, elementary proof. I noticed Shoup's proof in 2015 and realized that it could be adapted to give numeric bounds, which this report tries to realize. In fact, the main tools used in the arguments of this report are just a sharpening of the tools used in Shoup's proof, which helps to make the results exact instead of asymptotic.

The exact values of $\mu$ are rational numbers: expressible as a fractions with denominator $|S||R^S|$. I have computed the exact numerators for small values of $|R|$ and $|S|$. The last time I checked (2012), many of these numerators were not included in the OEIS.