Big-Key Symmetric Encryption: Resisting Key Exfiltration

Mihir Bellare[1(✉)], Daniel Kane[1], and Phillip Rogaway[2]

[1] Department of Computer Science and Engineering,
University of California, San Diego, USA
mihir@eng.ucsd.edu
[2] Department of Computer Science, University of California, Davis, USA
http://cseweb.ucsd.edu/~mihir/
http://cseweb.ucsd.edu/~dakane/
http://web.cs.ucdavis.edu/~rogaway/

Abstract. This paper aims to move research in the bounded retrieval model (BRM) from theory to practice by considering symmetric (rather than public-key) encryption, giving efficient schemes, and providing security analyses with sharp, concrete bounds. The threat addressed is malware that aims to exfiltrate a user's key. Our schemes aim to thwart this by using an enormously long key, yet paying for this almost exclusively in storage cost, not speed. Our main result is a general-purpose lemma, the *subkey prediction lemma*, that gives a very good bound on an adversary's ability to guess a (modest length) subkey of a big-key, the subkey consisting of the bits of the big-key found at random, specified locations, after the adversary has exfiltrated partial information about the big-key (e.g., half as many bits as the big-key is long). We then use this to design a new kind of key encapsulation mechanism, and, finally, a symmetric encryption scheme. Both are in the random-oracle model. We also give a less efficient standard-model scheme that is based on universal computational extractors (UCE). Finally, we define and achieve hedged BRM symmetric encryption, which provides authenticity in the absence of leakage.

1 Introduction

This paper is concerned with the possibility of mass surveillance by APTs. An APT (Advanced Persistent Threat) is malware that resides on your system and attempts to exfiltrate your key. (This means that it aims to communicate your key to its home base, probably using your system's network connection.) How might one protect against this? One answer is: by strengthening system security to the point that we eliminate APTs. Unfortunately, this approach seems out of reach. Indeed, the Snowden revelations show that the NSA (through TAO, their Tailored Access Operations unit) and others have sophisticated system penetration capabilities that they use to plant APTs. Another answer is provided by the bounded retrieval model (BRM) [2,3,17,20,23], namely to make secret keys so big that their undetected exfiltration is difficult.

M. Robshaw and J. Katz (Eds.): CRYPTO 2016, Part I, LNCS 9814, pp. 373–402, 2016.
DOI: 10.1007/978-3-662-53018-4_14

So far, BRM research has been largely theoretical and foundational. Our intent is to move it towards being a plausible countermeasure to mass surveillance. This involves the following. First, we treat symmetric rather than asymmetric encryption. Second, we focus on simple, efficient schemes. Third, we provide security analyses that are strong and fully concrete (no hidden constants), giving good numerical bounds on security.

Our main technical contribution is a very good upper bound on the probability of predicting a subset of random positions in a large key in the presence of leakage. We then give a big-key encapsulation mechanism, and thence a big-key symmetric encryption scheme, both efficient and in the random-oracle model (ROM) [12]. Let us now look at all this in more detail.

The BRM. The BRM evolved through a series of works [2,3,17,20,23]. It is part of the broader area of Leakage-Resilient Cryptography [1,25,34] and is also related to the Bounded Storage Model [16,29]. A survey by Alwen et al. [4] explains that

> If an attacker hacks into a remote system (or infects it with some malware) it may ... be infeasible/impractical for the attacker to download "too much" data (say, more than 10 GBytes).

In such a setting, making the key very big, say 1 TByte, ensures that the adversary obtains limited information about it. The idea was echoed by Adi Shamir at the RSA 2013 conference in a somewhat broader context of secrets that are not necessarily keys, and with specific reference to APTs. He said:

> We have to think in a totally different way about how we are going to protect computer systems assuming there are APTs inside already which cannot be detected. Is everything lost? I claim that not: there are many things that you can do, because the APT is basically going to have a very, very narrow pipeline to the outside world. ... I would like, for example, all the small data to become big data, just in terms of size. I want that the secret of the Coco-Cola company to be kept not in a tiny file of one kilobyte, which can be exfiltrated easily by an APT $\cdots$. I want that file to be a terabyte, which cannot be [easily] exfiltrated.

The problem we aim to solve is how to effectively utilize a key $\boldsymbol{K}$ whose length k is big in the presence of an adversary that has some information about $\boldsymbol{K}$. In the BRM, the APT is modeled as a function that takes $\boldsymbol{K}$ as input and returns a string L of $\ell < k$ bits, the *leakage*, where ℓ is some parameter; for example, $\ell = k/10$ corresponding to the assumption that the adversary can't exfiltrate more than 100 GBytes of information about a 1 TByte key. In the BRM, effective utilization imposes two requirements, one on security and the other on efficiency. The former is that security must be maintained in the presence of the leakage. The latter, called *locality*, is that the scheme's algorithms may access only a very small part of the key. Without this, working with a big-key would be too inefficient.

ADW [3] give BRM schemes for authenticated key-exchange and public-key identification. ADNSWW [2] give BRM schemes for public-key encryption. (Here $\boldsymbol{K}$ is the secret decryption key.) The latter construction is based

on identity-based hash proof systems, which are instantiated via bilinear maps, lattices and quadratic residuosity.

Overview. We treat symmetric encryption in the BRM. We refer to it, synonymously, as big-key symmetric encryption, to emphasize the use of a large key. The k-bit key $\boldsymbol{K}$ of a big-key symmetric encryption scheme $\mathbf{SE}$ is shared between sender and receiver. Algorithm $\mathbf{SE}.\mathsf{Enc}$ maps $\boldsymbol{K}$ and a plaintext to a ciphertext, while $\mathbf{SE}.\mathsf{Dec}$ maps $\boldsymbol{K}$ and a ciphertext to a plaintext, both making few accesses to $\boldsymbol{K}$.

Our scheme is simple, efficient, and easily described. It is parameterized by the number of probes p into the k-bit key $\boldsymbol{K}$, for example, $p = 500$. To encrypt message M, pick a random R, apply the random oracle RO to it to get a sequence of probes $\mathbf{p}[1], \ldots, \mathbf{p}[p] \in [1..k]$ into $\boldsymbol{K}$, and let $J = \boldsymbol{K}[\mathbf{p}[1]] \ldots \boldsymbol{K}[\mathbf{p}[p]]$ be the corresponding bits of $\boldsymbol{K}$. Next, obtain a short, conventional key K by applying RO to J. Finally, encrypt M under K with a conventional symmetric encryption scheme to get a ciphertext C. Return (R, C) as the ciphertext of the big-key scheme.

This scheme is derived via a modular framework with three steps, each involving its own definition, problem and analysis, namely *subkey prediction*, *encapsulation* and *encryption*. We will discuss them in turn below. Then we describe two extensions of the basic scheme sketched above, namely, a standard-model variant based on UCE, and the idea of *hedged* big-key encryption.

Subkey prediction. Our core technical contribution concerns the *subkey-prediction problem*. We consider a game parameterized by the length k of the big-key, a number p of random probes into it, and a bound ℓ on the leakage. The game selects a k-bit big-key $\boldsymbol{K}$ at random, and the adversary is given the result $L \in \{0,1\}^{\ell}$ of applying an arbitrary leakage function to $\boldsymbol{K}$. This L represents the exfiltrated information, and, as a special case, could consist of ℓ bits of $\boldsymbol{K}$. Now the game picks random indices $\mathbf{p}[1], \ldots, \mathbf{p}[p] \in [1..k]$, called probes. These probes are also given to the adversary. We ask the adversary, given the leakage and probes, to predict, in its entirety, the induced p-bit *subkey* $J = \boldsymbol{K}[\mathbf{p}[1]]\boldsymbol{K}[\mathbf{p}[2]] \cdots \boldsymbol{K}[\mathbf{p}[p]]$ found at those spots. We ask: how well can the adversary do at this? That is, we want to know the adversary's maximum probability, denoted $\mathsf{Adv}^{\mathrm{skp}}_{k,p,1}(\ell)$ in our formalization of Sect. 3, of guessing J as a function of k, ℓ, and p.

The analysis turns out to be surprisingly technical. One might think that there is no better strategy than to leak some ℓ bits of $\boldsymbol{K}$, for example the first ℓ, and then predict J in the obvious way. (If $\mathbf{p}[i] \in [1..\ell]$ then $\boldsymbol{K}[\mathbf{p}[i]]$ is known from the leakage, else guess it.) We give a counter-example showing that this is not the best strategy, and one can do better using an error-correcting code. We then show that, roughly, the best strategy for the adversary is to select the leakage function so that the pre-images of any point under this function are sandwiched between adjacent Hamming balls. In Sect. 3 we formalize and prove this and then use it to show that

$$\mathsf{Adv}^{\mathrm{skp}}_{k,p,1}(\ell) \approx 2^{-p \cdot w(\ell/k)} \quad \text{for} \quad w(\lambda) = -\lg(1 - H_2^{-1}(1-\lambda)) \ ,$$

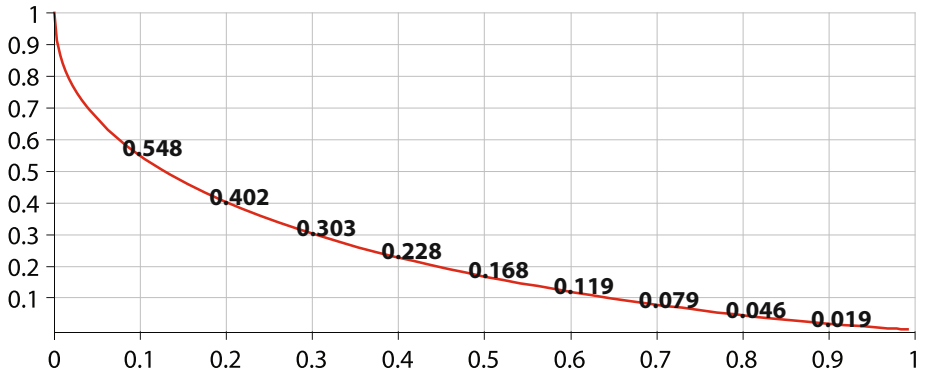

Fig. 1. Subkey predictability. The x-axis indicates the fraction $\lambda = \ell/k$ of the bits adversarially exfiltrated from the big-key. The corresponding point $w(\lambda)$ on the y-axis then indicates how many bits of unpredictability are achieved from each random probe (e.g., $w(0.5) \approx 0.168$). In particular, the adversary's ability to guess the contents of a p-bit probe are about $2^{-p \cdot w(\lambda)}$. Results apply to large k and modest p.

where $H_2(x) = -x \lg(x) - (1-x)\lg(1-x)$ is the binary entropy function and $H_2^{-1}(1-\lambda)$ is the smaller of the two possible inverses of $1-\lambda$ under H_2. The salient point is that the probability decreases exponentially in the number of probes p, with the factor in the exponent depending on the fraction $\lambda = \ell/k$ of the bits of the big-key that are leaked. See Fig. 1 for a plot of $w(\lambda)$ as a function of λ.

Related settings are analyzed by several lemmas in the literature, notably NZ [30, Lemma 11], Vadhan [35, Lemma 9] and ADW [3, Lemma A.3]. We are not aware of any direct way of applying the first two to get bounds on subkey prediction probability. (They do give bounds on what in Sect. 3 we call the restricted subkey prediction probability.) They also involve hidden constants that make it hard to obtain the concrete bounds needed to estimate security in usage. In contrast, the elegant lemma of ADW [3, Lemma A.3] can be directly applied to bound $\mathsf{Adv}^{\mathrm{skp}}_{k,p,1}(\ell)$, and it gives a concrete bound with no hidden constants. However, the bound obtained in this way is much inferior to ours, as we now illustrate.

In Sect. 3 we show that [3, Lemma A.3] implies $\mathsf{Adv}^{\mathrm{skp}}_{k,p,1}(\ell) \le 2^{-c}$ for c given by Eq. (8), namely $c = p(k-\ell-5)/(2k\lg(2k)+3p)$. Figure 2 compares the bounds obtained by our result (column "New") with the ones obtained via [3, Lemma A.3] (column "Old"). We see for example that for k being 1 TB and $\ell/k = 0.1$, for 500 probes, we show that $\mathsf{Adv}^{\mathrm{skp}}_{k,p,1}(\ell) \approx 2^{-274}$ while the prior bound would be only $2^{-5.1}$. Other entries show similarly large gaps for other parameter values. Another way to compare is, for a certain fixed k, ℓ, to ask how many probes are needed to get 256 bits of security, meaning have $\mathsf{Adv}^{\mathrm{skp}}_{k,p,1}(\ell) \le 2^{-256}$. According to Fig. 2, for k being 1 TB and $\ell/k = 0.1$, our result says that 468 probes suffice, while the prior result would require us to use 24954 probes. The difference in

p	New	Old
250	137	3.3
500	274	6.6
1000	548	13
234	**128**	3.1
468	**256**	6.2
9642	5284	**128**
19284	10567	**256**

1 GB, 10% leak
($k = 8 \cdot 10^9$, $\ell = 0.1k$)

p	New	Old
250	42	1.8
500	84	3.7
1000	168	7.4
762	**128**	5.6
1523	**256**	11
17536	2919	**128**
34711	5837	**256**

1 GB, 50% leak
($k = 8 \cdot 10^9$, $\ell = 0.5k$)

p	New	Old
250	137	2.6
500	274	5.1
1000	548	10
234	**128**	2.4
468	**256**	4.8
12477	6837	**128**
24954	13674	**256**

1 TB, 10% leak
($k = 8 \cdot 10^{12}$, $\ell = 0.5k$)

p	New	Old
250	42	1.4
500	84	2.8
1000	168	5.7
762	**128**	4.3
1523	**256**	8.7
22458	3777	**128**
44916	7553	**256**

1 TB, 50% leak
($k = 8 \cdot 10^{12}$, $\ell = 0.5k$)

Fig. 2. Numerical examples, comparisons with ADW [3, Lemma A.3]. The "New" and "Old" columns show approximate values of x for which $\mathsf{Adv}^{\mathrm{skp}}_{k,p,1}(\ell) \le 2^{-x}$ using our results and those of ADW [3, Lemma A.3], respectively. The first two tables use a key size k of 1 GB; the rest, 1 TB. In each case, we consider leakage restricted to 10 % or 50 % of the key length. The first column gives the number of probes p. In each table, the first three rows represent natural probe counts $p \in \{250, 500, 1000\}$ while the remaining four rows are determined by asking how many probes would be needed to get either 128-bit or 256-bit security according to each of the bounds.

efficiency is dramatic, meaning that reliance on the prior bounds would translate to a significant loss of practical efficiency for big-key symmetric encryption.

Encapsulation. Building on the above, we provide a general tool, XKEY, for using big-keys in symmetric settings. XKEY takes in a key $\boldsymbol{K}$ and a random *selector* R, which is a short string (like 128–256 bits). It returns a *derived key* $K = \mathsf{XKEY}(\boldsymbol{K}, R)$, which has conventional length (like 128 bits) and can be used in a conventional scheme. We formalize the goal of XKEY, which we call *big-key encapsulation*. It asks that the derived key is indistinguishable from a random string of the same length, even given the selector and leakage on the big-key. This is reminiscent of a classical key-encapsulation mechanism (KEM) as defined by CS [18], yet it is also very different, since we are in the presence of leakage and in the symmetric (rather than asymmetric) setting. Additionally, in the ROM, not only does the adversary have access to the random oracle RO, but, also, *the leakage function can also itself invoke the random oracle*. This is crucial, as otherwise it is easy to give an example of a scheme that is secure in the ROM yet insecure when the random oracle is instantiated. This element increases the technical difficulty of our security proof.

Given $\boldsymbol{K}$ and R, our XKEY algorithm applies the random oracle RO to R to specify probes $\mathbf{p}[1], \ldots, \mathbf{p}[p] \in [1..k]$ into the big-key $\boldsymbol{K}$. It lets $J = \boldsymbol{K}[\mathbf{p}[1]]\,\boldsymbol{K}[\mathbf{p}[2]] \cdots \boldsymbol{K}[\mathbf{p}[p]]$ be the corresponding subkey. By subkey unpredictability, J is unpredictable, but it is not guaranteed to be indistinguishable from random. XKEY further applies RO to $R \| J$ to obtain the derived key K. Theorem 12 says that this derived key is indistinguishable from random even to an adversary that sees multiple encapsulations and gets leakage about $\boldsymbol{K}$. The theorem gives

a concrete bound on the adversary advantage. The proof has two steps, first addressing the ability of the leakage function to use the random oracle by a coin-fixing argument, and then reducing to subkey prediction via a game sequence.

Big-key encryption. In Sect. 5, we define and achieve big-key symmetric encryption. Our definition ensures indistinguishability in the presence of leakage on the big-key, the leakage again allowed to depend on the random oracle. To encrypt a message M under $\boldsymbol{K}$, we pick R at random, obtain a session key $K = \mathsf{XKEY}(\boldsymbol{K}, R)$, and output as ciphertext a pair (R, C) where C is an encryption of M under K with a base, conventional symmetric encryption scheme, for example an AES mode of operation. Theorem 13 shows that this achieves our definition of big-key encryption privacy assuming that XKEY achieves our definition of encapsulation security and the base scheme meets a standard privacy definition for symmetric encryption. The scheme is very efficient. Relative to the base scheme, the added communications cost is small (transmission of R) and the added computation cost is also small (one XKEY operation).

Standard-model scheme. A variant of our scheme, still quite efficient, can be proven secure in the standard model. We can focus on encapsulation, since our reduction of encryption to the latter does not use a random oracle. We consider a variant $\mathsf{XKEY2}$ of XKEY where the selector $R = (I, \mathbf{p})$ is a key I for a UCE (Universal Computation Extractor) H [8] together with a sequence of probes $\mathbf{p}[1], \ldots, \mathbf{p}[p] \in [1..k]$ into $\boldsymbol{K}$. Given $\boldsymbol{K}$ and this selector, $\mathsf{XKEY2}$ lets $J = \boldsymbol{K}[\mathbf{p}[1]] \ldots \boldsymbol{K}[\mathbf{p}[p]]$ be the corresponding bits of $\boldsymbol{K}$ and obtains subkey K by applying $\mathsf{H}(I, \cdot)$ to J. Efficiency is the same as for our ROM scheme, but the ciphertext of the encryption scheme is longer because the selector, which is included in the ciphertext, is longer. Theorem 14 proves security assuming H is UCE$[\mathcal{S}^{\mathrm{sup}}]$-secure, namely UCE-secure for statistically unpredictable sources. This version of UCE, from [8,15], has been viable and has been used in many applications. We use our subkey unpredictability bound in a crucial way, to prove statistical unpredictability of the source constructed in the reduction.

Authenticity and hedged big-key encryption. Our big-key encryption schemes provide privacy. What about authenticity—meaning big-key authenticated encryption (AE)? ADW [3] remark that secure signatures are not possible in the BRM. The same attack applies to rule out big-key AE. Namely, the leakage function can simply compute and leak a valid ciphertext. We take the view that authenticity is important in normal usage but the target of mass surveillance is violating privacy, not authenticity. Accordingly, we suggest *hedged* big-key encryption. Encryption would continue to provide, in the presence of leakage, the guarantees of our above-discussed schemes. Additionally, in the absence of leakage, the same scheme should provide AE, meeting a standard and strong formalizations of the latter [10]. Throughout all this, the scheme must remain true to the local efficiency requirement of the BRM. In Sect. 7 we give a simple way to turn a privacy-only big-key scheme into a hedged one while preserving locality.

Discussion. We clarify some assumptions and limitations of the BRM and our work. In the BRM, leakage (exfiltration) on the big-key is assumed to occur once, at the beginning. The leakage function cannot depend on ciphertexts. This is true in most models of leakage-resilient cryptography, but leakage after encryption has been considered [26] and it would be interesting to extend this to big-key symmetric encryption. We assume encryption code is trusted. Algorithm substitution attacks [11] consider the case of untrusted encryption code. Whether any defense against ASAs is possible in the big-key setting remains open. Finally we assume the availability of trusted randomness in the encryption process.

Our schemes view the big-key as a string over $\{0,1\}$, so each probe draws one bit of the big-key. More generally, we could view the big-key as a vector over $\{0,1\}^b$ where b is some block or word length, for example $b = 8$ or $b = 32$. Each probe would then result in a b-bit string. This could increase efficiency and ease of implementation of the scheme. Our current subkey prediction lemma addresses only the $b = 1$ case. One can apply [3, Lemma A.3] to get a bound for larger b, but we would expect that an extension of our subkey prediction Lemma would yield better bounds. Obtaining such an extension is an interesting open question.

Related work. In Maurer's bounded storage model [29], parties have access to a public source of randomness that transmits a sequence $X_1, X_2, \ldots$ of high min-entropy strings. Parties are limited in storage and the goal is information theoretic security. Symmetric encryption in this setting is studied in [5,6,24,28,29,35]. However, in this information-theoretic setting one can derive only one session key from each output of the source and the ability to encrypt multiple messages relies on the expectation of receiving a continuous stream of strings from the source. In contrast, in the BRM setting, the big-key is static, and, in the presence of leakage on it, we want to encrypt an arbitrary number of messages without changing the key.

Di Crescenzo et al. [20] and Dziembowski [23] independently introduced the bounded retrieval model (BRM), where the adversary has a bounded amount of information on the data stored by the users. See the excellent survey of Alwen et al. [4] for history and results in this setting. Here we touch on only a few examples. DLW [20] design password protocols for the setting where the "password file" stored by the server is huge but the amount of information the adversary can get about it is limited, yet the server is efficient. Dziembowski [23] considers malicious code that can exfiltrate only a limited portion of a long key before it is sanitized. Dziembowski's aim is to achieve entity authentication and session-key distribution. Like us, the author works in the ROM. His symmetric key-derivation scheme, and its analysis, are similar to ours. Following up on this work, CDDLLW [17] provide a general paradigm for achieving intrusion-resilient authenticated key exchange (AKE), as well as a solution in the standard (as opposed to RO) model. Alwen et al. [3] design authenticated key agreement protocols in the public-key setting where the secret key is huge but the public key is small and security must be maintained in the presence of bounded leakage on

the secret key. ADNSWW [2] construct public-key encryption schemes in this model.

Predating the BRM, Kelsey and Schneier consider an authentication scheme in which a user with a large-memory token authenticates itself by providing XORs of randomly specified subsets of its bits [27]. An adversary who manages to exfiltrate only some of the bits of the device will be unable to subsequently impersonate the token. Dagon, Lee, and Lipton consider the problem of securely storing a ciphertext encrypted in a weak password on a device that's subject to adversarial attack [19]. They create a long ciphertext for a short plaintext where partial knowledge of the ciphertext will frustrate dictionary attacks.

Lu [28] and Vadhan [35] construct locally computable extractors. These yield big-key encapsulation schemes, but with the limitation that one can only obtain a small, bounded number of encapsulated keys from one big-key. (Encapsulated keys are statistically close to random, so after a few derivations, the entropy of the big-key is exhausted.) This is not sufficient for big-key symmetric encryption, where, with the big-key in place, we want to encrypt an arbitrary number of messages. XKEY in this light yields a locally computable *computational* extractor in the ROM. (The computational element is that the number of queries to the random oracle is limited. In asymptotic terms, it is a polynomial.) It uses the random oracle as a hardcore function following [12] to be able to encapsulate an unbounded number of keys under a single big-key. Our UCE-based encapsulation scheme XKEY2 similarly yields a standard-model locally-computable computational extractor. One might also view XKEY and XKEY2 as reusable locally-computable extractors following [2,17,23]. Reusability of extractors also aims to address deriving multiple subkeys and arose in [14,21,31].

A condenser [22,32,33] is a min-entropy extractor. Our subkey prediction Lemma can be viewed as building a BRM (or locally computable) condenser for a random source. The algorithm is to simply return the subkey J given by random probes into the big-key.

One could obtain a big-key symmetric encryption scheme by adapting the asymmetric BRM scheme of ADNSWW [2]. Our schemes are much more efficient.

2 Notation

NOTATION. For integers $a \le b$ we let $[a..b] = \{a, \ldots, b\}$. If $\mathbf{x}$ is a vector then $|\mathbf{x}|$ denotes its length and $\mathbf{x}[i]$ denotes its i-th coordinate. (For example if $\mathbf{x} = (10, 00, 1)$ then $|\mathbf{x}| = 3$ and $\mathbf{x}[2] = 00$.) We let ε denote the empty vector, which has length 0. If $0 \le i \le |\mathbf{x}|$ then we let $\mathbf{x}[1..i] = (\mathbf{x}[1], \ldots, \mathbf{x}[i])$, this being ε when $i = 0$. We let S^n denote the set of all length n vectors over the set S and we let S^* denote the set of all finite-length vectors over the set S. Strings are treated as the special case of vectors over $\{0,1\}$. Thus, if x is a string then $|x|$ is its length, $x[i]$ is its i-th bit, $x[1..i] = x[1]...x[i]$, ε is the empty string, $\{0,1\}^n$ is the set of n-bit strings and $\{0,1\}^*$ the set of all strings. If $\boldsymbol{K}$ is a k-bit string and $\mathbf{p}$ is a p-vector over $[1..k]$ then we let $\boldsymbol{K}[\mathbf{p}] = \boldsymbol{K}[\mathbf{p}[1]]\boldsymbol{K}[\mathbf{p}[2]] \cdots \boldsymbol{K}[\mathbf{p}[p]]$ denote the length-p string consisting of the bits of K in the positions indicated by $\mathbf{p}$. For example, if $K = 01010101$ and $\mathbf{p} = (1, 8, 2, 2)$ then $\boldsymbol{K}[\mathbf{p}] = 0111$.

If X is a finite set, we let $x \xleftarrow{} X$ denote picking an element of X uniformly at random and assigning it to x. Algorithms may be randomized unless otherwise indicated. Running time is worst case. If A is an algorithm, we let $y \leftarrow A(x_1, \cdots; r)$ denote running A with random coins r on inputs $x_1, \cdots$ and assigning the output to y. We let $y \xleftarrow{} A(x_1, \cdots)$ be the result of picking r at random and letting $y \leftarrow A(x_1, \cdots; r)$. We let $[A(x_1, \cdots)]$ denote the set of all possible outputs of A when invoked with inputs $x_1, \cdots$. We denote by $\mathrm{Func}[a, b]$ the set of all functions f from $\{0,1\}^a$ to $\{0,1\}^b$.

We use the code-based game-playing framework [13] (see Fig. 3 for an example). By $\Pr[\mathbf{G}]$ we denote the probability that game $\mathbf{G}$ returns true. Uninitialized boolean variables, sets and integers are assume initialized to false, the empty set and 0, respectively.

3 The Subkey Prediction Lemma

Suppose an adversary computes 1 TByte of information L about a random 2 TByte key $\boldsymbol{K}$. Afterward, we challenge the adversary to identify the 128-bit substring K whose bits are those found at some 128 random locations of $\boldsymbol{K}$. We tell the adversary those locations. How well can the adversary do at this game? This section introduces what we call the *subkey prediction* game to formalize this question and answer it.

The subkey-prediction game. Let $k, \ell, p, q \geq 1$ be integers with $k \geq \ell$. We call these values the *key length*, the *leakage length*, the *probe length*, and the *iteration count*. Let $\mathsf{Lk}\colon \{0,1\}^k \to \{0,1\}^\ell$ be a function, the leakage function. We associate to these values and an adversary $\mathcal{A}$ the *subkey prediction game* $\mathbf{G}^{\mathrm{skp}}_{k,p,q}(\mathcal{A}, \mathsf{Lk})$ depicted in the left panel of Fig. 3. (Ignore the other game for now). The game picks a random key $\boldsymbol{K}$ (the big-key) of length k and computes *leakage* $L \leftarrow \mathsf{Lk}(\boldsymbol{K})$. It then picks q random probes $\mathbf{p}_1, \ldots, \mathbf{p}_q$, each consisting of p random indexes into the key $\boldsymbol{K}$. The adversary $\mathcal{A}$ must guess one of the strings $\boldsymbol{K}[\mathbf{p}_i]$ given leakage L and probe locations $\mathbf{p}_1, \ldots, \mathbf{p}_q$. Any one will do. It outputs a guess J and the game returns true if and only if J is one of the values $\boldsymbol{K}[\mathbf{p}_i]$. The adversary needn't identify which subkey it has guessed. Now define

$$\begin{aligned}
\mathsf{Adv}^{\mathrm{skp}}_{k,p,q}(\mathcal{A}, \mathsf{Lk}) &= \Pr[\mathbf{G}^{\mathrm{skp}}_{k,p,q}(\mathcal{A}, \mathsf{Lk})] \\
\mathsf{Adv}^{\mathrm{skp}}_{k,p,q}(\mathsf{Lk}) &= \max_{\mathcal{A}} \mathsf{Adv}^{\mathrm{skp}}_{k,p,q}(\mathcal{A}, \mathsf{Lk}) \\
\mathsf{Adv}^{\mathrm{skp}}_{k,p,q}(\ell) &= \max_{\mathsf{Lk} \in \mathrm{Func}[k,\ell]} \mathsf{Adv}^{\mathrm{skp}}_{k,p,q}(\mathsf{Lk})
\end{aligned}$$

The first is the probability that the game returns true, that is, the probability that $\mathcal{A}$ wins the game. For the second definition the maximum is over all adversaries $\mathcal{A}$, regardless of running time. For the third definition the maximum is over all leakage functions $\mathsf{Lk}\colon \{0,1\}^k \to \{0,1\}^\ell$ that return ℓ bits. This last advantage function measures the best possible prediction probability when the

Game $\mathbf{G}^{\mathrm{skp}}_{k,p,q}(\mathcal{A},\mathsf{Lk})$	Game $\mathbf{G}^{\mathrm{skp1}}_{k,p}(\mathcal{A},\mathcal{K})$
$\boldsymbol{K} \twoheadleftarrow \{0,1\}^k$; $L \leftarrow \mathsf{Lk}(\boldsymbol{K})$	$\boldsymbol{K} \twoheadleftarrow \mathcal{K}$
for $i \leftarrow 1,\ldots,q$ do $\mathbf{p}_i \twoheadleftarrow [1..k]^p$	$\mathbf{p} \twoheadleftarrow [1..k]^p$
$J \leftarrow \mathcal{A}(L,\mathbf{p}_1,\ldots,\mathbf{p}_q)$	$J \leftarrow \mathcal{A}(\mathbf{p})$
return $\big(J \in \{\boldsymbol{K}[\mathbf{p}_1],\ldots,\boldsymbol{K}[\mathbf{p}_q]\}\big)$	return $(J = \boldsymbol{K}[\mathbf{p}])$

Fig. 3. Subkey-prediction game (left) and restricted subkey-prediction game (right).

leakage is restricted to ℓ bits and the adversary is arbitrary. We are interested in upper bounding this last advantage as a function of k, ℓ, p, q.

The idea here is that an adversary has some information about $\boldsymbol{K}$, but it is limited to $\mathsf{Lk}(\boldsymbol{K})$, which is ℓ bits. This leaves $k - \ell$ bits of average min-entropy in $\boldsymbol{K}$. We are trying to extract it in a particular and efficient way, by taking a random subset of positions of $\boldsymbol{K}$. We are asking "only" for unpredictability, and will then apply a random oracle to get random-looking bits.

From many queries to one. One simple observation is that we can reduce the case of q probes to the case of a single probe via the union bound:

Lemma 1. *Let k, ℓ, p, q be integers with $k \geq \ell$. Then*

$$\mathsf{Adv}^{\mathrm{skp}}_{k,p,q}(\ell) \leq q \cdot \mathsf{Adv}^{\mathrm{skp}}_{k,p,1}(\ell) \ \blacksquare$$

Proof (Lemma 1). Given adversary $\mathcal{A}$ we let $\mathcal{A}_1$ be the adversary defined as follows. On input $L, \mathbf{p}$ it picks $g \twoheadleftarrow [1..q]$ and $\mathbf{p}_j \twoheadleftarrow [1..k]^p$ for $j \in [1..q] \setminus \{g\}$. It lets $\mathbf{p}_g \leftarrow \mathbf{p}$ and returns $J \twoheadleftarrow \mathcal{A}(L, \mathbf{p}_1, \ldots \mathbf{p}_q)$. By a union bound we have $\mathsf{Adv}^{\mathrm{skp}}_{k,p,q}(\mathcal{A},\mathsf{Lk}) \leq q \cdot \mathsf{Adv}^{\mathrm{skp}}_{k,p,1}(\mathcal{A}_1,\mathsf{Lk})$ and the lemma follows. $\blacksquare$

We retain the q-probe definition because this is what we will use in applications, but Lemma 1 allows us to focus on the $q = 1$ case for the remainder of this section.

Connection to Hamming balls. Given k, ℓ, let us ask which leakage function $\mathsf{Lk}\colon \{0,1\}^k \to \{0,1\}^\ell$ maximizes the advantage. At first glance it's hard to imagine there's a strategy better than the greedy one of leaking the first ℓ bits of $\boldsymbol{K}$. Specifically let $\mathsf{Lk}_\ell\colon \{0,1\}^k \to \{0,1\}^\ell$ be defined by $\mathsf{Lk}_\ell(\boldsymbol{K}) = \boldsymbol{K}[1]\cdots\boldsymbol{K}[\ell]$, the function that returns the first ℓ bits of its input. Then a natural conjecture is that $\mathsf{Adv}^{\mathrm{skp}}_{k,p,1}(\ell) = \mathsf{Adv}^{\mathrm{skp}}_{k,p,1}(\mathsf{Lk}_\ell)$, meaning that the subkey prediction advantage is maximized by Lk_ℓ. Indeed this was our first guess.

This conjecture, however, is false. We now give a counter-example that shows this. Besides indicating the subtleties in the problem, it makes a connection with error-correcting codes and Hamming balls that will underly our eventual results and bound. Here is the counter-example. Let $k = 7$ and $\ell = 4$. Let $p = 1$. Then

$$\mathsf{Adv}^{\mathrm{skp}}_{k,1,1}(\mathsf{Lk}_\ell) = (4/7)(1) + (3/7)(1/2) = 11/14\ .$$

Now consider the following alternative Lk: $\{0,1\}^7 \to \{0,1\}^4$ for which we will show that $\mathsf{Adv}^{\mathrm{skp}}_{k,1,1}(\mathsf{Lk}) = 7/8 > 11/14$. Let $W_1, \ldots, W_{16} \in \{0,1\}^7$ be the codewords of the Hamming (7,4) code. (See Wikipedia article "Hamming(7,4)" for the definition.) This code has message length $\ell = 4$ and codeword length $k = 7$, so that it describes an (injective) encoding function E: $\{0,1\}^4 \to \{W_1, \ldots, W_{16}\} \subseteq \{0,1\}^7$. It has minimum distance 3 and corrects one error. Let B_i be the set of all 7-bit strings whose Hamming distance from W_i is ≤ 1. Then $|B_i| = 8$ and the sets $B_1, \ldots, B_{16}$ are a partition of $\{0,1\}^7$. The decoding function D: $\{0,1\}^7 \to \{0,1\}^4$, given $\boldsymbol{K}$, finds the unique i such that $\boldsymbol{K} \in B_i$. It then returns message $L = E^{-1}(W_i)$. The leakage function Lk is simply D; namely, given the big-key $\boldsymbol{K}$, it returns its decoding $D(\boldsymbol{K})$. Now the adversary $\mathcal{A}$ receives (L, p_1) where $p_1 \in [1..7]$ is the random probe into $\boldsymbol{K}$ chosen by the game and $L = \mathsf{Lk}(\boldsymbol{K}) = D(\boldsymbol{K})$, and it wants to predict $\boldsymbol{K}[p_1]$. Adversary $\mathcal{A}$ lets $W = E(L)$ and returns $W[p_1]$ as its guess. Then $\mathsf{Adv}^{\mathrm{skp}}_{k,1,1}(\mathcal{A}, \mathsf{Lk}) = 7/8$ because if $W = W_i$ then 7 of the 8 strings in B_i have $W_i[p_1]$ as their p_1-th bit. So

$$\mathsf{Adv}^{\mathrm{skp}}_{k,1,1}(\mathsf{Lk}) \geq 7/8 > 11/14 = \mathsf{Adv}^{\mathrm{skp}}_{k,1,1}(\mathsf{Lk}_\ell) \; .$$

Subkey-prediction bound. We now give our upper bound on $\mathsf{Adv}^{\mathrm{skp}}_{k,p,1}(\ell)$ as a function of k, ℓ, p. Let $w_{\mathrm{H}}(x)$ denote the Hamming weight of string x. Let $\mathcal{B}_k(r) = \{\, x \in \{0,1\}^k : w_{\mathrm{H}}(x) \leq r \,\}$ be the Hamming ball of radius r with center 0^k and let $B_k(r) = |\mathcal{B}_k(r)|$ be its size. Then

$$B_k(r) = \sum_{i=0}^{r} \binom{k}{i} \; . \tag{1}$$

Now define the *radius* $\mathsf{rd}_k(N)$ as the largest integer r such that $B_k(r) \leq N$, and let

$$\begin{aligned} & G_{k,p}(N) \\ & = \frac{1}{N} \sum_{i=0}^{\mathsf{rd}_k(N)} \binom{k}{i} \left(1 - \frac{i}{k}\right)^p + \frac{N - B_k(\mathsf{rd}_k(N))}{N} \left(1 - \frac{1 + \mathsf{rd}_k(N)}{k}\right)^p \qquad (2) \\ & \leq \frac{1}{N} \sum_{i=0}^{\min(1+\mathsf{rd}_k(N),k)} \binom{k}{i} \left(1 - \frac{i}{k}\right)^p \; . \qquad (3) \end{aligned}$$

The following says this function provides an upper bound on the subkey prediction probability:

Theorem 2 (Subkey-prediction bound). *Let k, ℓ, p be integers, $k \geq \ell$. Then*

$$\mathsf{Adv}^{\mathrm{skp}}_{k,p,1}(\ell) \leq G_{k,p}(2^{k-\ell}) \; \blacksquare \tag{4}$$

Before we prove Theorem 2, let's try to understand the growth rate of the bound for parameters of interest.

ESTIMATES. The $G_{k,p}(N)$ formula itself is somewhat intractable. To use Theorem 2 it is easier to work with the following approximation that we used in Sect. 1. The approximation is very good. For $\lambda \in [0,1]$ we let

$$w(\lambda) = -\lg(1 - H_2^{-1}(1-\lambda))$$

where lg is the logarithm to base two, H_2 is the binary entropy function defined for $x \in [0,1]$ by $H_2(x) = -x\ln(x) - (1-x)\ln(1-x)$ and $H_2^{-1}(1-\lambda)$ returns the smaller of the two values x satisfying $H_2(x) = 1-\lambda$. Then for $\ell \le k$ we have

$$G_{k,p}(2^{k-\ell}) \approx 2^{-p\cdot w(\ell/k)} \ . \tag{5}$$

We now justify this. Let us define

$$G^*_{k,p,r}(N) = \frac{1}{N}\sum_{i=0}^{r} \binom{k}{i}\left(1-\frac{i}{k}\right)^p \ . \tag{6}$$

The following lemma gives both a lower bound and an upper bound on $G^*_{k,p,r}(N)$ that are within a factor two of each other, showing that the estimate of the lemma is tight to within a small constant. The proof is in [9].

Lemma 3. *Let $N, k, r, p \ge 1$ be integers with $r \le k$. Let $\theta = r/k$ and $\gamma = 2\binom{k}{r}/N$. Suppose: (1) $p \le 0.27 \cdot k$ and (2) $\theta \le 0.22$. Let $\alpha = -\lg(1-\theta) > 0$. Then*

$$\frac{\gamma}{2}\cdot 2^{-\alpha\cdot p} \le G^*_{k,p,r}(N) \le \gamma \cdot 2^{-\alpha\cdot p} \ \blacksquare$$

Letting $r^* = \mathsf{rd}_k(2^{k-\ell})$, from Eq. (3) we have $G_{k,p}(2^{k-\ell}) \approx G^*_{k,p,r^*}(2^{k-\ell})$, which by Lemma 3 is $\approx 2^{-\alpha\cdot p}$ where $\alpha = -\lg(1-r^*/k)$. So to justify Eq. (5) we need to show that $\alpha \approx w(\ell/k)$. For this we use the well-known estimate $2^{k-\ell} \approx B_k(r^*) \approx 2^{k\cdot H_2(r^*/k)}$ to get $1-\ell/k \approx H_2(r^*/k)$ or $r^*/k \approx H_2^{-1}(1-\ell/k)$. Thus $-\lg(1-r^*/k) \approx -\lg(1-H_2^{-1}(1-\ell/k)) = w(\ell/k)$ as desired.

RESTRICTED SUBKEY-PREDICTION. Our proof of Theorem 2 will rely on an analysis of the *restricted* subkey-prediction game $\mathbf{G}^{\mathrm{skp1}}_{k,p}(\mathcal{A},\mathcal{K})$ shown in the right panel of Fig. 3. The game is associated to integers k, p, adversary $\mathcal{A}$ and a set $\mathcal{K} \subseteq \{0,1\}^k$. Our results concerning this game are also of independent interest because we obtain bounds that are *tight*. In this game, the big-key $\boldsymbol{K}$ is drawn from $\mathcal{K}$ rather than from $\{0,1\}^k$, and there is no leakage. Also, there is only one probe. The rest is the same as in the subkey prediction game. Intuitively, one can think of $\mathcal{K}$ as being $\mathsf{Lk}^{-1}(L)$ for some particular L received by $\mathcal{A}$ in game $\mathbf{G}^{\mathrm{skp}}$, so that the new game, $\mathbf{G}^{\mathrm{skp1}}$, effectively represents the view of $\mathcal{A}$ in the prior game at the point it receives leakage L. Now define

$$\begin{aligned}
\mathsf{Adv}^{\mathrm{skp1}}_{k,p}(\mathcal{A},\mathcal{K}) &= \Pr[\mathbf{G}^{\mathrm{skp1}}_{k,p}(\mathcal{A},\mathcal{K})] \\
\mathsf{Adv}^{\mathrm{skp1}}_{k,p}(\mathcal{K}) &= \max_{\mathcal{A}} \mathsf{Adv}^{\mathrm{skp1}}_{k,p}(\mathcal{A},\mathcal{K}) \\
\mathsf{Adv}^{\mathrm{skp1}}_{k,p}(N) &= \max_{\mathcal{K}\subseteq\{0,1\}^k,\ |\mathcal{K}|=N} \mathsf{Adv}^{\mathrm{skp1}}_{k,p}(\mathcal{K})
\end{aligned}$$

The first advantage is the probability that $\mathcal{A}$ wins the game. In the second, the maximum is over all adversaries $\mathcal{A}$, regardless of running time. In the third, the maximum is over all sets $\mathcal{K} \subseteq \{0,1\}^k$ that have size $|\mathcal{K}| = N$.

Main lemmas and proof of theorem. Theorem 2 is obtained via two main lemmas. Here we state them and show how they yield the theorem. We will then prove the lemmas. The first main lemma reduces the task of upper bounding the subkey prediction probability $\mathsf{Adv}^{\mathrm{skp}}_{k,p,1}(\ell)$ to the task of bounding the special subkey prediction probability $\mathsf{Adv}^{\mathrm{skp1}}_{k,p}(N)$ for $N = 2^{k-\ell}$:

Lemma 4. *Let $k, \ell, p \geq 1$ be integers with $k \geq \ell$. Then*

$$\mathsf{Adv}^{\mathrm{skp}}_{k,p,1}(\ell) \;\leq\; \mathsf{Adv}^{\mathrm{skp1}}_{k,p}(2^{k-\ell}) \;\blacksquare \tag{7}$$

The proof of this lemma, given below, involves a definition of concavity for discrete functions and a lemma saying where such functions attain their maximum. Then a particular function $F_{k,p}$ we define is shown to meet this definition of concavity, and Lemma 4 results. The second main lemma characterizes the special subkey prediction probability:

Lemma 5. *Suppose $1 \leq N \leq 2^k$ and $p \geq 1$. Then*

$$\mathsf{Adv}^{\mathrm{skp1}}_{k,p}(N) = G_{k,p}(N) \;\blacksquare$$

We note that Lemma 5 is an equality, not a bound. We are able to say *exactly* what is the special subkey prediction probability for a given value of N. Lemma 5 is obtained by showing that for a given N, the maximum of $\mathsf{Adv}^{\mathrm{skp1}}_{k,p}(\mathcal{K})$ over sets $\mathcal{K}$ of size N, occurs for a set that is *monotone* and sandwiched in between two adjacent Hamming balls. Monotone means that if a string is in the set, so is any string obtained by flipping one bits to zero bits on the original string. For monotone sets, it is quite easy to estimate the optimal advantage. All this put together will lead to Lemma 5.

The proof of Theorem 2 is immediate from these two lemmas, which we still need to prove. But first, with the above, we are in a position to compare with prior work.

Comparison with prior work. Lemmas related to subkey and restricted subkey prediction have been given by NZ [30, Lemma 11], Vadhan [35, Lemma 9] and ADW [3, Lemma A.3]. Briefly, the first two don't give bounds on subkey prediction. They do give bounds on restricted subkey prediction but they are hard to use due to hidden constants, and these bound are inferior to Lemma 4 since the latter is tight. The elegant lemma of ADW [3, Lemma A.3], however, not only applies directly to subkey prediction but also gives a concrete bound with no hidden constants. The difference here, as quantified in Sect. 1, is that the bound is much inferior to that of Theorem 2, translating to a significant loss of practical efficiency for big-key symmetric encryption.

NZ [30, Lemma 11] considers drawing a string $\boldsymbol{K}$ from $\{0,1\}^k$ according to a distribution D with min-entropy δ. Like us, for a random probe $\mathbf{p} \in [1..k]^p$, they then consider $\boldsymbol{K}[\mathbf{p}]$. The lemma specifies ϵ, δ' such that for all but an ϵ fraction of the $\mathbf{p}$'s, the distribution $\boldsymbol{K}[\mathbf{p}]$ is within statistical distance ϵ of a δ' source. Unlike our work there is no leakage. Their setting does not capture subkey prediction but it does capture restricted subkey prediction game, which corresponds to the distribution D that puts a uniform probability on $\mathcal{K}$ and 0 probability on points outside it. However, the formulas in [30, Lemma 11] make ϵ very large for the parameters of interest to us, and also have un-specified constants. In contrast Lemma 5 gives a tight bound with no unspecified constants. The same remains true with Vadhan [35, Lemma 9]. (Here the hidden constant is an exponent in the statistical distance.) The values of the constants can in principle, of course, be obtained from the proofs, but since our bound of Lemma 4 for restricted subkey prediction is tight, we would not see an improvement. (And the indication from what follows, where concrete bounds are given, is that the bounds would be substantially worse than ours anyway.)

ADW [3, Lemma A.3] can, however, be directly applied to get a bound on $\mathsf{Adv}^{\mathrm{skp}}_{k,p,1}(\ell)$. Referring to their lemma, let random variable X represent the big-key $\boldsymbol{K}$, and let experiment $\mathcal{E}_1$ represent the leakage. Their t corresponds to our p. (We think the N in their lemma is a typo. It should be t.) Then $\tilde{\mathrm{H}}_\infty(X) = k - \ell$. The lemma then implies that $\mathsf{Adv}^{\mathrm{skp}}_{k,p,1}(\ell) \le 2^{-c}$ as long as $k - \ell \ge 2ck \lg(2k)/p + 3c + 5$. This translates to

$$c \le \frac{p(k - \ell - 5)}{2k \lg(2k) + 3p} \,. \tag{8}$$

This is the formula used for Fig. 2.

We note that ADW [3, Lemma A.3] considers a more general setting than subkey prediction. We are saying that in this special setting, we can get much better bounds. In the big-key context, an important case where their bound applies but ours does not is when we work over blocks rather than bits. Here $\boldsymbol{K}$ is a k-vector over $\{0,1\}^b$ for some block length parameter b, so that each probe draws a b-bit block. This setting is more convenient for implementations of big-key symmetric encryption. Giving better bounds for this setting is an interesting open question.

Proof of Lemma 5. Recall that $w_{\mathrm{H}}(x)$ denotes the Hamming weight of string x. For equal-length strings $x, y \in \{0,1\}^*$, write $x \preceq y$ if $x[i] \le y[i]$ —that is, if $x[i] = 1$ then $y[i] = 1$— for every $i \in [1..|x|]$. Write $x \prec y$ if $x \preceq y$ and $x \ne y$. Then $\preceq$ is a partial order on $\{0,1\}^k$, satisfying the required conditions, namely it is reflexive, anti-symmetric and transitive. If $U \subseteq \{0,1\}^*$ is a finite set then we let $w_{\mathrm{H}}(U) = \sum_{u \in U} w_{\mathrm{H}}(u)$ be the Hamming weight of U. Call a subset $\mathcal{K} \subseteq \{0,1\}^k$ monotone if for all $x, y \in \{0,1\}^k$ we have that if $x \prec y$ and $y \in \mathcal{K}$ then $x \in \mathcal{K}$. The following lemma says that the maximum restricted subkey prediction advantage occurs for a set $\mathcal{K}$ that is monotone. The proof is in [9].

Lemma 6. *Suppose* $1 \le N \le 2^k$ *and* $p \ge 1$. *Then there is a monotone* $\mathcal{K} \subseteq \{0,1\}^k$ *such that* $|\mathcal{K}| = N$ *and*

$$\mathsf{Adv}^{\mathrm{skp1}}_{k,p}(N) = \mathsf{Adv}^{\mathrm{skp1}}_{k,p}(\mathcal{K}) \ \blacksquare$$

Now define the function $g_{k,p}$, taking input a set $\mathcal{K} \subseteq \{0,1\}^k$, via

$$g_{k,p}(\mathcal{K}) = \frac{1}{|\mathcal{K}|} \sum_{x \in \mathcal{K}} \left(1 - \frac{w_{\mathrm{H}}(x)}{k}\right)^p . \tag{9}$$

A nice property of monotone sets is that we can easily compute the maximal advantage on them, via the function just defined:

Lemma 7. *Suppose* $1 \le N \le 2^k$ *and* $p \ge 1$. *Suppose* $\mathcal{K} \subseteq \{0,1\}^k$ *is a monotone set of size* N. *Then*

$$\mathsf{Adv}^{\mathrm{skp1}}_{k,p}(\mathcal{K}) = g_{k,p}(\mathcal{K}) \ \blacksquare$$

Proof (Lemma 7). Since $\mathcal{K}$ is monotone, the best strategy for the adversary given $\mathbf{p}$ is to simply return $K = 0^k$ as the guess. Letting $\mathcal{A}$ denote the adversary that does this, we now want to evaluate $\mathsf{Adv}^{\mathrm{skp1}}_{k,p}(\mathcal{A}, \mathcal{K})$. This is just the probability that if x is chosen at random from $\mathcal{K}$ and $\mathbf{p}[1], \ldots, \mathbf{p}[p]$ are chosen at random from $[1..k]$ then $x[\mathbf{p}[j]] = 0$ for all $j \in [1..p]$. This probability is $g_{k,p}(\mathcal{K})$. $\blacksquare$

We will now further characterize the sets which attain the maximum. We say that $\mathcal{K} \subseteq \{0,1\}^k$ is sandwiched between Hamming balls if there is an r such that $\mathcal{B}_k(r) \subseteq \mathcal{K} \subset \mathcal{B}_k(r+1)$. Note in this case it must be that $r = \mathsf{rd}_k(N)$ where N is the size of $\mathcal{K}$. The proof of the following, which exploits Lemma 6, is in [9].

Lemma 8. *Suppose* $1 \le N \le 2^k$ *and* $p \ge 1$. *Let* $r = \mathsf{rd}_k(N)$. *Then there is a monotone, size* N *set* $\mathcal{K}$ *such that* $\mathcal{B}_k(r) \subseteq \mathcal{K} \subset \mathcal{B}_k(r+1)$ *and*

$$\mathsf{Adv}^{\mathrm{skp1}}_{k,p}(N) = \mathsf{Adv}^{\mathrm{skp1}}_{k,p}(\mathcal{K}) \ \blacksquare$$

We are now in a position to prove our second main lemma.

Proof (Lemma 5). Let $r = \mathsf{rd}_k(N)$. By Lemma 8 there is a monotone, size N set $\mathcal{K}$ such that $\mathcal{B}_k(r) \subseteq \mathcal{K} \subset \mathcal{B}_k(r+1)$ and $\mathsf{Adv}^{\mathrm{skp1}}_{k,p}(N) = \mathsf{Adv}^{\mathrm{skp1}}_{k,p}(\mathcal{K})$. But

$$\mathsf{Adv}^{\mathrm{skp1}}_{k,p}(\mathcal{K}) = g_{k,p}(\mathcal{K}) = G_{k,p}(N)$$

where the first equality is by Lemma 7 and the second is because $\mathcal{B}_k(r) \subseteq \mathcal{K} \subset \mathcal{B}_k(r+1)$. $\blacksquare$

Proof of Lemma 4. We now prove the first main lemma. We begin with a general result about the maximization of discrete concave functions.

The standard definition of concavity of functions applies to continuous functions. Here we provide a definition for functions defined on a discrete domain that allows us to prove a lemma we will use later. Proceeding, suppose $F\colon \mathbb{Z}_M \to \mathbb{R}$. We say that F is concave if $F(a+1)-F(a) \le F(b+1)-F(b)$ for all $a, b \in \mathbb{Z}_{M-1}$ satisfying $a \ge b$. Now suppose $t, m \ge 1$ are integers with $1 \le m \le t$. Then we let

$$S(M,m,t) = \{\ (x_1,\ldots,x_m) \in \mathbb{Z}_M^m :\ x_1 + \cdots + x_m = t\ \}\ .$$

Define $F^m\colon \mathbb{Z}_M^m \to \mathbb{R}$ by $F^m(x_1,\ldots,x_m) = F(x_1) + \cdots + F(x_m)$. The proof of the following is in [9].

Lemma 9. *Suppose $F\colon \mathbb{Z}_M \to \mathbb{R}$ is concave. Suppose $1 \le m \le t$ are integers such that m divides t and $t/m \in \mathbb{Z}_M$. Then*

$$\max_{(x_1,\ldots,x_m)\in S(M,m,t)} F^m(x_1,\ldots,x_m) = m \cdot F(t/m)\ \blacksquare$$

That is, the maximum of F^m over $S(M,m,t)$ is attained when all inputs have the same value $t/m \in \mathbb{Z}_M$, and is thus equal to $m \cdot F(t/m)$. To apply this in our setting, we introduce the function $F_{k,p}\colon \mathbb{Z}_{2^k+1} \to \mathbb{R}$ defined by

$$F_{k,p}(N) = \frac{N}{2^k} \cdot \mathsf{Adv}^{\mathrm{skp1}}_{k,p}(N)\ . \tag{10}$$

Rewriting the claim of Lemma 4 in terms of the function $F_{k,p}$, our aim is to show that $\mathsf{Adv}^{\mathrm{skp}}_{k,p,1}(\ell) \le 2^\ell \cdot F_{k,p}(2^{k-\ell})$. The following says that the function $F_{k,p}$ meets our definition of concavity. The proof is in [9].

Lemma 10. *Let $k, p \ge 1$ be integers. Then the function $F_{k,p}$ is concave.* $\blacksquare$

We now show how to obtain our first main lemma.

Proof (Lemma 4). Let $M = 2^k + 1$, $m = 2^\ell$ and $t = 2^k$. Now we have

$$\begin{aligned}
\mathsf{Adv}^{\mathrm{skp}}_{k,p,1}(\ell) &= \max_{\mathsf{Lk}} \left(\sum_L \frac{|\mathsf{Lk}^{-1}(L)|}{2^k} \cdot \max_{\mathcal{A}} \Pr[\ \mathbf{G}^{\mathrm{skp}}_{k,p,1}(\mathcal{A},\mathsf{Lk}) \mid \mathsf{Lk}(\boldsymbol{K}) = L\ ] \right) \\
&= \max_{\mathsf{Lk}} \left(\sum_L \frac{|\mathsf{Lk}^{-1}(L)|}{2^k} \cdot \mathsf{Adv}^{\mathrm{skp1}}_{k,p}(\mathsf{Lk}^{-1}(L)) \right) \\
&\le \max_{(N_1,\ldots,N_m)\in S(M,m,t)} \sum_{i=1}^{m} F_{k,p}(N_i) \\
&= \max_{(N_1,\ldots,N_m)\in S(M,m,t)} F^m_{k,p}(N_1,\ldots,N_m) \\
&= m \cdot F_{k,p}(2^{k-\ell}) \qquad (11) \\
&= m \cdot \frac{2^{k-\ell}}{2^k} \cdot \mathsf{Adv}^{\mathrm{skp1}}_{k,p}(2^{k-\ell}) = \mathsf{Adv}^{\mathrm{skp1}}_{k,p}(2^{k-\ell})\ . \qquad (12)
\end{aligned}$$

Equation (11) is justified as follows. Lemma 10 says that $F_{k,p}$ is concave. So we can apply Lemma 9, and here $t/m = 2^{k-\ell}$. Equation (12) is by Eq. (10) and because $m = 2^{\ell}$. ▮

4 Encapsulating a Key

In this section we introduce *big-key encapsulation*. A scheme for this aim lets a user encapsulate a random, conventional-length key K using a big-key $\boldsymbol{K}$. We speak of "encapsulation" instead of "encryption" because the user never selects a value K to encrypt: rather, a random R is chosen and this value, together with the big-key $\boldsymbol{K}$, determines a derived key $K = \mathsf{KEY}(\boldsymbol{K}, R)$. A user can transmit R to a party that knows $\boldsymbol{K}$ and in this way name an induced key K. While the aim is similar to a KEM (key encapsulation mechanism) [18], there are also many differences, which we will later discuss.

Definitions. A big-key encapsulation algorithm is a deterministic algorithm KEY that, given strings $\boldsymbol{K} \in \{0,1\}^k$ and $R \in \{0,1\}^r$, returns a string $K = \mathsf{KEY}(\boldsymbol{K}, R) \in \{0,1\}^\kappa$. We call $\boldsymbol{K}$, R, and K the *big-key* , *selector*, and *derived key*, respectively. Their lengths, being part of the signature of KEY, are numbers associated to it. Since we will be working in the RO model, we allow the encapsulation algorithm to depend on an oracle RO. We write such a function as a superscript, $K = \mathsf{KEY}^{\mathrm{RO}}(\boldsymbol{K}, R)$, if we want to emphasize its presence.

The security requirement for an encapsulation algorithm captures the idea that a derived key K should be indistinguishable from a uniform random string even when accompanied by R and some bounded amount of leakage from the big-key $\boldsymbol{K}$. This is formalized via the game $\mathbf{G}^{\mathrm{key}}_{\mathsf{KEY}}(\mathcal{A})$ on the left of Fig. 4. The game is associated to encapsulation algorithm KEY and adversary $\mathcal{A}$. The game is in the ROM, oracle RO taking (x, l) and returning a random l-bit string. Not only do algorithms and the adversary have access to RO, but, importantly, so does the leakage function $\mathsf{LK}^{\mathrm{RO}}\colon \{0,1\}^k \to \{0,1\}^\kappa$, now called an *oracle* leakage function to emphasize this.

In its first stage, the adversary specifies the leakage function it wants. It then makes a sequence of Derive calls, each providing an (R, K) pair that is either *real*—the derived key was determined by running KEY with a random selector—or *random*—the derived key is uniformly random. Which of these possibilities occurs depends on a bit b chosen at the beginning of the game. The adversary must guess that bit. We let $\mathsf{Adv}^{\mathrm{key}}_{\mathsf{KEY}}(\mathcal{A}) = 2\Pr[\mathbf{G}^{\mathrm{key}}_{\mathsf{KEY}}(\mathcal{A})] - 1$ be its advantage in doing so.

Discussion. A big-key encapsulation algorithm is in some ways similar to a conventional key encapsulation mechanism [18]. But there are many differences. First, we are in the symmetric setting instead of the asymmetric setting. Second, we are considering security under leakage, so the leakage function, chosen by the adversary, comes into the picture. Finally, we have chosen a syntax under which we do not have a separate decapsulation algorithm, preferring to surface the coins R across the encapsulation algorithm's interface.

Game $\mathbf{G}^{\text{key}}_{\mathsf{KEY}}(\mathcal{A})$	DERIVE( )	RO(x, l)
$b \twoheadleftarrow \{0,1\}$	$R \twoheadleftarrow \{0,1\}^r$	if not $T[x,l]$ then
$\boldsymbol{K} \twoheadleftarrow \{0,1\}^k$	if $(b = 0)$ then $K \twoheadleftarrow \{0,1\}^\kappa$	$\quad T[x,l] \twoheadleftarrow \{0,1\}^l$
$(\mathsf{LK}, \sigma) \twoheadleftarrow \mathcal{A}^{\text{RO}}( )$	else $K \leftarrow \mathsf{KEY}^{\text{RO}}(\boldsymbol{K}, R)$	return $T[x,l]$
$L \leftarrow \mathsf{LK}^{\text{RO}}(\boldsymbol{K})$	return (R, K)	
$b' \twoheadleftarrow \mathcal{A}^{\text{DERIVE, RO}}(L, \sigma)$		
return $(b' = b)$		

Fig. 4. Game for defining the security of a big-key key encapsulation algorithm $\mathsf{KEY}\colon \{0,1\}^k \times \{0,1\}^r \to \{0,1\}^\kappa$

One may ask why we let the adversary encapsulate an arbitrary number of conventional keys, using its DERIVE oracle q times; wouldn't a hybrid argument show that providing a single derived key is equivalent, up to a factor of q? In fact, such a hybrid argument fails because the single-query adversary has no means to simulate the real derived keys. For this reason, it is important to consider multiple keys.

We explain the importance of giving the leakage function access to RO. Otherwise, the scheme $\mathsf{KEY}^{\text{RO}}(\boldsymbol{K}, R) = \text{RO}(\boldsymbol{K}, \kappa)$ is secure. Yet, in practice, when RO is instantiated with a concrete hash function $H\colon \{0,1\}^* \to \{0,1\}^\kappa$, this scheme is certainly not secure because the leakage function can return $H(\boldsymbol{K})$. (We note that in this example the scheme does not make efficient use of $\boldsymbol{K}$ as we want, but this is an orthogonal issue to security.) Once the leakage function has access to RO, it too can return $\text{RO}(\boldsymbol{K}, \kappa)$, preventing this and other similar schemes from being deemed secure.

ENCAPSULATION SCHEME XKEY. Let $k, \kappa, r \geq 1$ be integers, and, for convenience, assume k is a power of two. Given the results of Sect. 3, a natural big-key encapsulation algorithm is as follows. On input the big-key $\boldsymbol{K} \in \{0,1\}^k$, algorithm KEY picks random probes $\mathbf{p} \in [1..k]^p$ and computes the induced subkey $J = \boldsymbol{K}[\mathbf{p}]$. While this subkey is unpredictable, up to the bounds we have seen, it is not indistinguishable from random bits, so cannot, by itself, function as the derived key K. So, instead, the algorithm would let the derived key be $K \leftarrow \text{RO}(J, \kappa)$ where κ is the desired length for the derived key.

While the above might sound reasonable, there are two problems with the scheme—one with regard to security and the other with regard to efficiency. We explain these and then present a scheme that resolves them.

The security problem is that, we can only show security for the scheme just described if the leakage function does not have access to the oracle RO. But we have argued that it *must* have such access, and in that case the proof breaks down and it is not clear if the scheme is secure. A simple remedy is to have the scheme pick a random string and include it in the scope of the RO used to determine the subkey. The proof can then exploit the fact that leakage function can't depend on this (not yet chosen) value.

This still leaves the efficiency issue, which is that the probe $\mathbf{p}$ is quite long, a total of $p \lg(k)$ bits. The number of components p of $\mathbf{p}$ is fairly large and grows with the fraction of bits potentially leaked; it will typically be 100–1000. If $\boldsymbol{K}$ is 1 TByte then $k \approx 2^{43}$ and we're using up a few kilobytes to communicate the selector $\mathbf{p}$, which is unpleasant.

Since we're working in the ROM, an easy solution is to obtain $\mathbf{p}$ by applying RO to a short seed. Conveniently, the same random choices needed for this remedy can be used for the security problem as well. The resulting encapsulation algorithm $\mathsf{XKEY}_{k,\kappa,p,r}\colon \{0,1\}^k \times \{0,1\}^r \to \{0,1\}^\kappa$ is shown in Fig. 5 and described below.

$$\begin{array}{l}\underline{\text{Algorithm } \mathsf{XKEY}^{\mathrm{RO}}_{k,\kappa,p,r}(\boldsymbol{K}, R)} \\ \text{for } i \leftarrow 1, \ldots, p \text{ do } \mathbf{p}[i] \leftarrow \mathrm{RO}(\langle R, i, 0\rangle,\ \lg(k)) \\ J \leftarrow \boldsymbol{K}[\mathbf{p}]\ ;\ K \leftarrow \mathrm{RO}(\langle R, J, 1\rangle,\ \kappa)\ ;\ \text{Return } K\end{array}$$

Fig. 5. Encapsulation algorithm XKEY. Given a length-k big-key $\boldsymbol{K}$ and a length-r selector R, the algorithm returns a length-κ subkey K. The value p, a security parameter, specifies the number of probes into the big-key.

The XKEY encapsulation algorithm picks or is provided a random selector R of length r. It picks the ith probe into $\boldsymbol{K}$ not directly at random, but by applying the random oracle RO to a string encoding R, i, and 0. In defining $\mathbf{p}[i]$ we interpret a $\lg(k)$-bit string as an integer in $[1..k]$ in the natural way. The encapsulation algorithm then lets the subkey J be the positions of $\boldsymbol{K}$ indicated by the probes. The derived key K is obtained by applying the RO to a string encoding J, R, and 0. The third component in each encoding $\langle \cdot, \cdot, \cdot \rangle$ is for domain separation.

Theorem 12 will establish security of XKEY, providing concrete bounds. Those bounds indicate that $r = |R|$ can be chosen to be quite small, like 128–256. Since only this value is transmitted when XKEY is used, bandwidth overhead in small and independent of p.

Enhanced subkey-prediction game. Towards the proof of Theorem 12 it is convenient to consider an enhanced version of the subkey-prediction problem. The security goal reflects two changes over our earlier subkey-prediction game $\mathbf{G}^{\mathrm{skp}}$. First, the leakage function is not fixed but dynamically chosen by an adversary. Second, that choice may depend on the RO, which the leakage function itself may depend on. These issues, particularly the second, lead to design choices embedded in XKEY and make proofs more challenging. We must now revisit sample predictability, formulate it in the extended setting just discussed, and then show that security in this new setting is implied by security in the basic one. Afterwards, we will be in a position to prove security for XKEY.

As in the basic sample predictability problem, let $k, \ell, p, q \geq 1$ be integers with $\ell \leq k$. Consider game $\mathbf{G}^{\mathrm{skp2}}_{k,p,q}(\mathcal{B})$ defined in Fig. 6. In its first stage, the adversary returns an oracle leakage function Lk. We say that $\mathcal{B}$ leaks (or exfiltrates) ℓ bits if LK: $\{0,1\}^k \to \{0,1\}^\ell$. Adversary $\mathcal{B}$ also returns state σ representing information that could be known to LK and will be passed to the adversary's second stage. The game picks the big-key $\boldsymbol{K}$, computes the leakage as $\mathsf{LK}^{\mathrm{RO}}(\boldsymbol{K})$ and picks probe vectors $\mathbf{p}_1, \ldots, \mathbf{p}_q$. These implicitly determine subkeys $K_1, \ldots, K_q$ where $K_i = \boldsymbol{K}[\mathbf{p}_i]$. In its second stage, the adversary $\mathcal{B}$ gets the leakage L and probe vectors as before, but now additionally gets σ. Its task is to guess one of the subkeys, and the game returns true if it does this correctly. We let $\mathsf{Adv}^{\mathrm{skp2}}_{k,p,q}(\mathcal{B}) = \Pr[\mathbf{G}^{\mathrm{skp2}}_{k,p,q}(\mathcal{B})]$ be its advantage.

Game $\mathbf{G}^{\mathrm{skp2}}_{k,p,q}(\mathcal{B})$	$\mathrm{RO}(x,l)$
$(\mathsf{LK}, \sigma) \twoheadleftarrow \mathcal{B}^{\mathrm{RO}}$	if not $T[x,l]$ then $T[x,l] \twoheadleftarrow \{0,1\}^l$
$\boldsymbol{K} \twoheadleftarrow \{0,1\}^k$; $L \leftarrow \mathsf{LK}^{\mathrm{RO}}(\boldsymbol{K})$	return $T[x,l]$
for $i \leftarrow 1, \ldots, q$ do $\mathbf{p}_i \twoheadleftarrow [1..k]^p$	
$K \twoheadleftarrow \mathcal{B}^{\mathrm{RO}}(L, \mathbf{p}_1, \ldots, \mathbf{p}_q, \sigma)$	
return $\big(K \in \{\boldsymbol{K}[\mathbf{p}_1], \ldots, \boldsymbol{K}[\mathbf{p}_p]\}\big)$	

Fig. 6. Game defining the "enhanced" subkey-prediction game, $\mathbf{G}^{\mathrm{skp2}}$. The game differs from $\mathbf{G}^{\mathrm{skp}}$ by allowing the adversary to select LK, both the latter and the adversary having access to a random oracle.

Lemma 11. *Let $\ell, k, p, q \geq 1$ be integers with $\ell \leq k$. Let $\mathcal{B}$ be an adversary leaking ℓ bits. Then*

$$\mathsf{Adv}^{\mathrm{skp2}}_{k,p,q}(\mathcal{B}) \leq \mathsf{Adv}^{\mathrm{skp}}_{k,p,q}(\ell) \quad \blacksquare \tag{13}$$

The proof uses a fairly standard "coin-fixing" argument in which a predictor adversary uses the "best" choice of random oracle and coins for $\mathcal{B}$. The details follow.

Proof (Lemma 11). Let $\mathbf{H}$ denote the set of all functions H such that $H(\cdot, l)$: $\{0,1\}^* \to \{0,1\}^l$ for all $l \in \mathbb{N}$. A random oracle is a function drawn at random from $\mathbf{H}$. Regard $\mathcal{B}, \ell, k, p, q$ as fixed and define the function g: $\mathbf{H} \times \{0,1\}^* \to [0,1]$ as $g(H, \omega) = \Pr[\mathbf{G}(H, \omega)]$ where game $\mathbf{G}(H, \omega)$ is on the left below:

Game $\mathbf{G}(H, \omega)$	Adversary $\mathcal{P}(L, \mathbf{p}_1, \ldots, \mathbf{p}_q)$
$(\mathsf{LK}, \sigma) \leftarrow \mathcal{B}^H(\omega)$; $\boldsymbol{K} \twoheadleftarrow \{0,1\}^k$	$J' \leftarrow \mathcal{B}^{H^*}(L, \mathbf{p}_1, \ldots, \mathbf{p}_q, \sigma^*; \omega^*)$
$L \leftarrow \mathsf{LK}^H(\boldsymbol{K})$	return J'
for $j \leftarrow 1, \ldots, q$ do	
$\quad \mathbf{p}_j[1], \ldots, \mathbf{p}_j[p] \twoheadleftarrow [1..k]$	
$\quad J_j \leftarrow \boldsymbol{K}[\mathbf{p}_j]$	
$J' \leftarrow \mathcal{B}^H(L, \mathbf{p}_1, \ldots, \mathbf{p}_q, \sigma; \omega)$	
return $(J' \in \{J_1, \ldots, J_q\})$	

In this game, the coins of $\mathcal{B}$ are fixed to ω and its random oracle is fixed to H. The probability is only over the choices made in the game. Let $(H^*, \omega^*) \in \mathbf{H} \times \{0,1\}^*$ be such that $g(H^*, \omega^*) \geq g(H, \omega)$ for all $(H, \omega) \in \mathbf{H} \times \{0,1\}^*$, and let $(\mathsf{LK}, \sigma^*) \leftarrow \mathcal{B}^{H^*}(\omega^*)$. Let $\mathsf{Lk} = \mathsf{LK}^{H^*}$. This is a basic (not oracle) leakage function, $\mathsf{Lk}\colon \{0,1\}^k \to \{0,1\}^\ell$. Define predictor adversary $\mathcal{P}$ as on the right above. Then

$$\begin{aligned}\mathsf{Adv}^{\mathrm{skp2}}_{k,p,q}(\mathcal{B}) &= \mathbf{E}_{(H,\omega) \twoheadleftarrow \mathbf{H}\times\{0,1\}^*}\left[g(H,\omega)\right] \\ &\leq g(H^*, \omega^*) \\ &= \mathsf{Adv}^{\mathrm{skp}}_{k,p,q}(\mathcal{P}, \mathsf{Lk}) \leq \mathsf{Adv}^{\mathrm{skp}}_{k,p,q}(\ell)\end{aligned}$$

which yields Eq. (13). ▌

Theorem 12. *Let $k, \kappa, p, r \geq 1$ be integers with k a power of two. Let $\mathsf{KEY} = \mathsf{XKEY}_{k,\kappa,p,r}$ be the big-key encapsulation scheme associated to them as per Fig. 5. Let $\mathcal{A}$ be an adversary making at most q queries to its* DERIVE *oracle and leaking ℓ bits. Assume the number of* RO *queries made by $\mathcal{A}$ in its first stage, plus the number made by the oracle leakage function LK that it outputs in this stage, is at most q_1, and the number of* RO *queries made by $\mathcal{A}$ in its second stage is at most q_2. Then*

$$\mathsf{Adv}^{\mathrm{key}}_{\mathsf{KEY}}(\mathcal{A}) \;\leq\; q_2 \cdot \mathsf{Adv}^{\mathrm{skp}}_{k,p,q}(\ell) + \frac{q \cdot (2q_1 + q - 1)}{2^{r+1}} \;\blacksquare \tag{14}$$

Game $\boxed{\mathbf{G}_0}$, $\mathbf{G}_1$

$\boldsymbol{K} \twoheadleftarrow \{0,1\}^k$
for $j \leftarrow 1, \ldots, q$ do
  $\mathbf{R}[j] \twoheadleftarrow \{0,1\}^r$; $\mathbf{p}_j[1], \ldots, \mathbf{p}_j[p] \twoheadleftarrow [1..k]$; $\mathbf{KK}[j] \twoheadleftarrow \{0,1\}^\kappa$; $J_j \leftarrow \boldsymbol{K}[\mathbf{p}_j]$
  For $i = 1, \ldots, j-1$ do
    If $(\mathbf{R}[j] = \mathbf{R}[i])$ then $(\mathbf{p}_j, J_j) \leftarrow (\mathbf{p}_i, J_i)$; $\mathsf{bad} \leftarrow \mathsf{true}$; $\boxed{\mathbf{KK}[j] \leftarrow \mathbf{KK}[i]}$
$(\mathsf{LK}, \sigma) \twoheadleftarrow \mathcal{A}^{\mathrm{RO}}$; $L \leftarrow \mathsf{LK}^{\mathrm{RO}}(\boldsymbol{K})$; $c' \twoheadleftarrow \mathcal{A}^{\textsc{Derive},\mathrm{RO}}(L, \sigma)$; Return $(c' = 1)$

Game $\mathbf{G}_2$, $\mathbf{G}_3$, $\mathbf{G}_4$, $\mathbf{G}_5$

$\boldsymbol{K} \twoheadleftarrow \{0,1\}^k$
for $j \leftarrow 1, \ldots, q$ do
  $\mathbf{R}[j] \twoheadleftarrow \{0,1\}^r$; $\mathbf{p}_j[1], \ldots, \mathbf{p}_j[p] \twoheadleftarrow [1..k]$; $\mathbf{KK}[j] \twoheadleftarrow \{0,1\}^\kappa$; $J_j \leftarrow \boldsymbol{K}[\mathbf{p}_j]$
  For $i = 1, \ldots, j-1$ do
    If $(\mathbf{R}[j] = \mathbf{R}[i])$ then $(\mathbf{p}_j, J_j) \leftarrow (\mathbf{p}_i, J_i)$
stage $\leftarrow 1$; $(\mathsf{LK}, \sigma) \twoheadleftarrow \mathcal{A}^{\mathrm{RO}}$; $L \leftarrow \mathsf{LK}^{\mathrm{RO}}(\boldsymbol{K})$
stage $\leftarrow 2$; $j \leftarrow 0$; $c' \twoheadleftarrow \mathcal{A}^{\textsc{Derive},\mathrm{RO}}(L, \sigma)$; Return $(c' = 1)$

DERIVE()
$j \leftarrow j + 1$; return $(\mathbf{R}[j], \mathbf{KK}[j])$

Fig. 7. Games for proof of Theorem 12. See Fig. 8 for the RO procedures.

$\underline{\mathrm{RO}(x,l)}$ // Games $\mathbf{G}_0$, $\mathbf{G}_1$
if not $T[x,l]$ then
  $T[x,l] \twoheadleftarrow \{0,1\}^l$
  for $j \leftarrow 1,\ldots,q$ do
    for $i \leftarrow 1,\ldots,p$ do
      if $(x = (\mathbf{R}[j], i, 0)$ and $l = \lg(k))$ then $T[x,l] \leftarrow \mathbf{p}_j[i]$
    if $(x = (\mathbf{R}[j], J_j, 1)$ and $l = r)$ then $T[x,l] \leftarrow \mathbf{KK}[j]$
return $T[x,l]$

$\underline{\mathrm{RO}(x,l)}$ // Games $\boxed{\mathbf{G}_2}$, $\mathbf{G}_3$
if not $T[x,l]$ then
  $T[x,l] \twoheadleftarrow \{0,1\}^l$
  if (stage $= 1$) then
    for $j \leftarrow 1,\ldots,q$ do
      for $i \leftarrow 1,\ldots,p$ do
        if $(x = (\mathbf{R}[j], i, 0)$ and $l = \lg(k))$ then $\mathsf{bad} \leftarrow \mathsf{true}$; $\boxed{T[x,l] \leftarrow \mathbf{p}_j[i]}$
      if $(x = (\mathbf{R}[j], J_j, 1)$ and $l = r)$ then $\mathsf{bad} \leftarrow \mathsf{true}$; $\boxed{T[x,l] \leftarrow \mathbf{KK}[j]}$
  if (stage $= 2$) then
    for $j \leftarrow 1,\ldots,q$ do
      for $i \leftarrow 1,\ldots,p$ do
        if $(x = (\mathbf{R}[j], i, 0)$ and $l = \lg(k)))$ then $T[x,l] \leftarrow \mathbf{p}_j[i]$
      if $(x = (\mathbf{R}[j], J_j, 1)$ and $l = r)$ then $T[x,l] \leftarrow \mathbf{KK}[j]$
return $T[x,l]$

$\underline{\mathrm{RO}(x,l)}$ // Games $\boxed{\mathbf{G}_4}$, $\mathbf{G}_5$
if not $T[x,l]$ then
  $T[x,l] \twoheadleftarrow \{0,1\}^l$
  if (stage $= 2$) then
    for $j \leftarrow 1,\ldots,q$ do
      for $i \leftarrow 1,\ldots,p$ do
        if $(x = (\mathbf{R}[j], i, 0)$ and $l = \lg(k))$ then $T[x,l] \leftarrow \mathbf{p}_j[i]$
      if $(x = (\mathbf{R}[j], J_j, 1)$ and $l = r)$ then $\mathsf{bad} \leftarrow \mathsf{true}$; $\boxed{T[x,l] \leftarrow \mathbf{KK}[j]}$
return $T[x,l]$

Fig. 8. RO procedures for games for proof of Theorem 12.

We note that the bound of Eq. (14) does not depend on the length κ of the output keys.

Proof (Theorem 12). Consider the games of Fig. 7. Their RO procedures are in Fig. 8. Game $\mathbf{G}_0$ includes the boxed code and is equivalent to case of game $\mathbf{G}^{\mathrm{key}}_{\mathsf{KEY}}(\mathcal{A})$ in which $b = 1$. Game $\mathbf{G}_5$ excludes the boxed code and mimics the $b = 0$ case of game $\mathbf{G}^{\mathrm{key}}_{\mathsf{KEY}}(\mathcal{A})$, except that the probability the former returns true

is the probability the latter returns false. Let $p_i = \Pr[\mathbf{G}_i]$ for $0 \le i \le 5$. Then

$$\begin{aligned}
&\mathsf{Adv}^{\mathrm{key}}_{\mathsf{KEY}}(\mathcal{A}) \\
&= \Pr[\, \mathbf{G}^{\mathrm{key}}_{\mathsf{KEY}}(\mathcal{A}) \mid b = 1 \,] - \Big(1 - \Pr[\, \mathbf{G}^{\mathrm{key}}_{\mathsf{KEY}}(\mathcal{A}) \mid b = 0 \,]\Big) \\
&= \Pr[\mathbf{G}_0] - \Pr[\mathbf{G}_5] \\
&= (p_0 - p_1) + (p_1 - p_2) + (p_2 - p_3) + (p_3 - p_4) + (p_4 - p_5) \\
&= (p_0 - p_1) + (p_2 - p_3) + (p_4 - p_5) \qquad (15) \\
&\le \Pr[\mathbf{G}_1 \text{ sets } \mathsf{bad}] + \Pr[\mathbf{G}_3 \text{ sets } \mathsf{bad}] + \Pr[\mathbf{G}_5 \text{ sets } \mathsf{bad}] \,. \qquad (16)
\end{aligned}$$

Equation (15) used the fact that $p_1 = p_2$ and $p_3 = p_4$. Equation (16) used the Fundamental Lemma of Game Playing [13]. Now

$$\Pr[\mathbf{G}_1 \text{ sets } \mathsf{bad}] \le \frac{q(q-1)}{2^{r+1}} \,. \qquad (17)$$

Also

$$\Pr[\mathbf{G}_3 \text{ sets } \mathsf{bad}] \le \frac{q \cdot q_1}{2^r} \,. \qquad (18)$$

The first estimate of the above bound may be that each RO query can set the first instance of bad with probability $pq/2^r$ and the second with probability $q/2^r$

Adversary $\mathcal{B}^{\mathrm{RO}}$
$(\mathsf{LK}, \sigma) \twoheadleftarrow \mathcal{A}^{\mathrm{RO}}$
return $(\mathsf{LK}, (\sigma, T))$

ROSIM(x, l)
If not T[x,l] then
  $T[x, l] \leftarrow \mathrm{RO}(x, l)$
Return $T[x, l]$

Adversary $\mathcal{B}^{\mathrm{RO}}(L, \mathbf{p}_1, \ldots, \mathbf{p}_q, (\sigma, T))$
for $j \leftarrow 1, \ldots, q$ do
  $\mathbf{R}[j] \twoheadleftarrow \{0,1\}^r$; $\mathbf{KK}[j] \twoheadleftarrow \{0,1\}^\kappa$
  for $i = 1, \ldots, j-1$ do
    If $(\mathbf{R}[j] = \mathbf{R}[i])$ then $\mathbf{p}_j \leftarrow \mathbf{p}_i$
$S \leftarrow \emptyset$; $j \leftarrow 0$; $c' \twoheadleftarrow \mathcal{A}^{\textsc{Derive},\mathrm{ROSIM}}(L, \sigma)$
$J' \twoheadleftarrow S$; return J'

Derive()
$j \leftarrow j + 1$; return $(\mathbf{R}[j], \mathbf{KK}[j])$

ROSIM(x, l)
If not $T[x, l]$ then
  $T[x, l] \leftarrow \mathrm{RO}(x, l)$
  for $j \leftarrow 1, \ldots, q$ do
    for $i \leftarrow 1, \ldots, p$ do
      if $(x = (\mathbf{R}[j], i, 0)$ and $l = \lg(k))$ then
        $T[x, l] \leftarrow \mathbf{p}_j[i]$
$(R, J, b) \leftarrow x$
If $b = 1$ then $S \leftarrow S \cup \{J\}$
return $T[x, l]$

Fig. 9. Adversary $\mathcal{B}$ for proof of Theorem 12.

for a bound of $q_1(p+1)q/2^r$. But these different events are mutually exclusive due to the queries including the index i and the domain separation bit, whence Eq. (18). Now we will present an adversary $\mathcal{A}$ such that

$$\Pr[\mathbf{G}_5 \text{ sets } \mathsf{bad}] \leq q_2 \cdot \mathsf{Adv}^{\mathrm{skp2}}_{k,p,q}(\mathcal{B}) \ . \tag{19}$$

The theorem follows from Lemma 11. Adversary $\mathcal{B}$ is shown in Fig. 9. In the first stage, $\mathcal{B}$ simulates $\mathcal{A}$'s RO directly via its own RO, keeping track of values in table T, which is passed to the next stage. In the second it does the same, makes sure to map selectors to probes as per game $\mathbf{G}_5$, and also it saves subkey guesses in the set S. Equation (19) follows because $|S| \leq q_2$. ▌

5 Big-Key Symmetric Encryption

Here we define and achieve big-key symmetric encryption.

Definitions. A symmetric encryption scheme SE specifies a key length $\mathsf{SE.kl} \in \mathbb{N}$, an encryption algorithm $\mathsf{SE.Enc}$ that given a key K and message M returns a ciphertext, and a deterministic decryption algorithm $\mathsf{SE.Dec}$ such that for all K, M we have $\mathsf{SE.Dec}(K, \mathsf{SE.Enc}(K, M)) = M$ with probability one, where the probability is over the coins of $\mathsf{SE.Enc}$. Privacy is formalized by left or right indistinguishability [7] via game $\mathbf{IND}_{\mathsf{SE}}(\mathcal{A})$ on the right of Fig. 10 associated to SE and adversary $\mathcal{A}$. We let $\mathsf{Adv}^{\mathrm{ind}}_{\mathsf{SE}}(\mathcal{A}) = 2\Pr[\mathbf{IND}_{\mathsf{SE}}(\mathcal{A})] - 1$ be the advantage of $\mathcal{A}$ in violating privacy of SE.

Syntactically, a big-key symmetric encryption scheme $\mathbf{SE}$ continues to be a symmetric encryption scheme as above, specifying $\mathbf{SE}.\mathsf{kl}$, $\mathbf{SE}.\mathsf{Enc}$ and $\mathbf{SE}.\mathsf{Dec}$. Two things make it special. First, privacy is measured under leakage on the key. Second, the encryption and decryption algorithms have the "locality" efficiency attribute, which means that in any one execution they access only a small part of the key. Privacy is formalized via game $\mathbf{LIND}_{\mathbf{SE}}(\mathcal{A})$ on the left of Fig. 10 associated to $\mathbf{SE}$ and adversary $\mathcal{A}$. In its first stage, the adversary, given access to RO, specifies an oracle leakage function $\mathsf{LK}\colon \{0,1\}^{\mathbf{SE}.\mathsf{kl}} \to \{0,1\}^{\ell}$ together with state information σ. We refer to ℓ as the number of bits leaked by $\mathcal{A}$. In its second stage, $\mathcal{A}$ gets the leakage $L \leftarrow \mathsf{LK}^{\mathrm{RO}}(\boldsymbol{K})$, the state σ, and access to the challenge encryption oracle Enc, while continuing to have access to RO. To win it must guess the challenge bit b. We let $\mathsf{Adv}^{\mathrm{lind}}_{\mathbf{SE}}(\mathcal{A}) = 2\Pr[\mathbf{LIND}_{\mathbf{SE}}(\mathcal{A})] - 1$ be the advantage of $\mathcal{A}$ in violating privacy of $\mathbf{SE}$ under leakage. Locality will not be formalized but rather visible in specific constructs. Giving the leakage function access to RO is important for same reason as we discussed in Sect. 4 for key encapsulation, namely that, otherwise, there are trivial ROM schemes that are secure but when the RO is instantiated the resulting scheme is clearly not secure.

Big-key encryption scheme. Let SE be a symmetric encryption scheme. Let KEY be a key-encapsulation algorithm with big-key length k, randomness length r and derived key length $\kappa = \mathsf{SE.kl}$ (keys output by KEY are suitable for use with SE). We associate to $\mathsf{SE}, \mathsf{KEY}$ the big-key symmetric encryption scheme

Game $\mathbf{LIND}_{\mathsf{SE}}(\mathcal{A})$	Game $\mathbf{IND}_{\mathsf{SE}}(\mathcal{A})$
$(\mathsf{LK}, \sigma) \twoheadleftarrow \mathcal{A}^{\mathrm{RO}}$; $\boldsymbol{K} \twoheadleftarrow \{0,1\}^{\mathbf{SE}.\mathsf{kl}}$	$S \twoheadleftarrow \{0,1\}^{\mathsf{SE}.\mathsf{kl}}$
$L \leftarrow \mathsf{Lk}^{\mathrm{RO}}(\boldsymbol{K})$; $b \twoheadleftarrow \{0,1\}$	$b \twoheadleftarrow \{0,1\}$
$c' \leftarrow \mathcal{A}^{\mathrm{Enc},\mathrm{RO}}(L, \sigma)$	$c' \leftarrow \mathcal{A}^{\mathrm{Enc}}$
Return $(c' = b)$	Return $(c' = b)$
$\underline{\mathrm{Enc}(M_0, M_1)}$	$\underline{\mathrm{Enc}(M_0, M_1)}$
$\overline{C} \twoheadleftarrow \mathbf{SE}.\mathsf{Enc}^{\mathrm{RO}}(\boldsymbol{K}, M_b)$	$C \twoheadleftarrow \mathsf{SE}.\mathsf{Enc}(S, M_b)$
Return $\overline{C}$	Return C
$\underline{\mathrm{RO}(x, l)}$	
If not $T[x,l]$ then $T[x,l] \twoheadleftarrow \{0,1\}^l$	
Return $T[x,l]$	

Fig. 10. Game defining privacy of symmetric encryption scheme $\mathbf{SE}$ under leakage, and game defining standard privacy of symmetric encryption scheme SE.

$\mathbf{SE} = \mathsf{BKSE}[\mathsf{SE}, \mathsf{KEY}]$ defined as follows. The key length is $\mathbf{SE}.\mathsf{kl} = k$ (the key for $\mathbf{SE}$ is the same as that for KEY) and the encryption and decryption algorithms are as follows:

Algorithm $\mathbf{SE}.\mathsf{Enc}^{\mathrm{RO}}(\boldsymbol{K}, M)$	Algorithm $\mathbf{SE}.\mathsf{Dec}^{\mathrm{RO}}(\boldsymbol{K}, \overline{C})$
$R \twoheadleftarrow \{0,1\}^r$; $K \twoheadleftarrow \mathsf{KEY}^{\mathrm{RO}}(\boldsymbol{K}, R)$	$(R, C) \leftarrow \overline{C}$
$C \twoheadleftarrow \mathsf{SE}.\mathsf{Enc}(K, M)$; $\overline{C} \leftarrow (R, C)$	$K \leftarrow \mathsf{KEY}^{\mathrm{RO}}(\boldsymbol{K}, R)$
Return $\overline{C}$	$M \leftarrow \mathsf{SE}.\mathsf{Dec}(K, C)$
	Return M

Encryption applies the key encapsulation algorithm to the big-key to get a derived key K. The message is encrypted under SE using key K. The locality of this scheme is exactly that of KEY since accesses to the key are done only by KEY. The big-key aspect is similarly inherited from KEY. The following says that our big-key scheme achieves privacy under leakage on the key assuming standard privacy of the base scheme SE and the lror-security of KEY. The proof is in [9].

Theorem 13. *Let* SE *be a symmetric encryption scheme. Let* KEY *be a key encapsulation algorithm with big-key length* k*, randomness length* r *and derived key length* $\kappa = \mathsf{SE}.\mathsf{kl}$*. Let* $\mathbf{SE} = \mathsf{BKSE}[\mathsf{SE}, \mathsf{KEY}]$ *be the big-key symmetric encryption scheme associated to them as above. Let* $\mathcal{A}$ *be an adversary making at most* q *queries to its* Enc *oracle and leaking* ℓ *bits. Then the proof below specifies an adversary* $\mathcal{A}_1$ *and an adversary* $\mathcal{A}_2$ *such that*

$$\mathsf{Adv}^{\mathrm{lind}}_{\mathbf{SE}}(\mathcal{A}) \;\le\; \mathsf{Adv}^{\mathrm{key}}_{\mathsf{KEY}}(\mathcal{A}_2) + q \cdot \mathsf{Adv}^{\mathrm{ind}}_{\mathsf{SE}}(\mathcal{A}_1)\ . \tag{20}$$

Adversary $\mathcal{A}_1$ *makes only one query to its* Enc *oracle and its running time is about that of* $\mathcal{A}$*. Adversary* $\mathcal{A}_2$ *makes* q *queries to its* Derive *oracle and has*

running time about that of $\mathcal{A}$. Its first stage is the same as that of $\mathcal{A}$, so it also leaks ℓ bits. In its second stage it makes the same number of RO *queries as $\mathcal{A}$ does in its second stage.* ▌

Since $\mathcal{A}_1$ makes only one query to its ENC oracle, a one-time encryption scheme suffices to instantiate SE.

6 Standard-Model Big-Key Encryption

In this section we give a standard-model variant of our scheme whose security relies on UCE. The scheme is as efficient as our ROM one, but ciphertexts are longer.

DEFINITIONS. We recall the UCE framework following [8]. Let $\mathsf{H}\colon \{0,1\}^{\mathsf{H.kl}} \times \{0,1\}^{\mathsf{H.il}} \to \{0,1\}^{\mathsf{H.ol}}$ be a family of functions taking an $\mathsf{H.kl}$-bit key I and $\mathsf{H.il}$-bit input x to a $\mathsf{H.ol}$-bit output $\mathsf{H}(I,x)$. Game $\mathbf{G}^{\mathrm{uce}}$ of Fig. 11 is associated to H, an adversary $\mathcal{S}$ called the *source*, an adversary $\mathcal{D}$ called the distinguisher, and a number q of keys. (We are using the multi-key version of UCE from [8].) Here $\mathcal{S}$ does not get the keys. It produces leakage M that is passed to $\mathcal{D}$, who does get the keys. We let $\mathsf{Adv}^{\mathrm{uce}}_{\mathsf{H},q}(\mathcal{S},\mathcal{D}) = 2\Pr[\mathbf{G}^{\mathrm{uce}}_{\mathsf{H},q}(\mathcal{S},\mathcal{D})] - 1$. We can only expect this to be small for sources restricted in some way. We require statistical unpredictability [8,15] of the source's oracle queries. If $\mathcal{P}$ is an adversary called the *predictor*, let $\mathsf{Adv}^{\mathrm{pred}}_{\mathcal{S}}(\mathcal{P}) = \Pr[\mathbf{G}^{\mathrm{sp}}_{\mathcal{S}}(\mathcal{P})]$ where the game is again in Fig. 11, and let $\mathsf{Adv}^{\mathrm{pred}}_{\mathcal{S}} = \max_{\mathcal{P}} \mathsf{Adv}^{\mathrm{pred}}_{\mathcal{S}}(\mathcal{P})$. The predictor here is unbounded (corresponding to statistical unpredictability) so the maximum is over all predictors. The assumption, informally, is that if $\mathsf{Adv}^{\mathrm{pred}}_{\mathcal{S}}$ is small then so is $\mathsf{Adv}^{\mathrm{uce}}_{\mathsf{H},q}(\mathcal{S},\mathcal{D})$ for all efficient $\mathcal{S},\mathcal{D}$. An important element of results is thus to be able to bound $\mathsf{Adv}^{\mathrm{pred}}_{\mathcal{S}}$ for the $\mathcal{S}$ constructed by the reduction.

XKEY2. The encapsulation algorithm is specified in Fig. 12. If κ is the desired length of the derived key, k the length of the big-key $\boldsymbol{K}$ (assumed a power of two for simplicity) and p the number of probes then it uses a family of functions

Game $\mathbf{G}^{\mathrm{uce}}_{\mathsf{H},q}(\mathcal{S},\mathcal{D})$	Game $\mathbf{G}^{\mathrm{sp}}_{\mathcal{S}}(\mathcal{P})$
$b \twoheadleftarrow \{0,1\}$	$Q \leftarrow \emptyset$; $M \twoheadleftarrow \mathcal{S}^{\mathrm{HASH}}$; $x \twoheadleftarrow \mathcal{P}(M)$
For $i = 1,\ldots,q$ do $I_i \twoheadleftarrow \{0,1\}^{\mathsf{H.kl}}$	Return $(x \in Q)$
$M \twoheadleftarrow \mathcal{S}^{\mathrm{HASH}}$; $b' \twoheadleftarrow \mathcal{D}(I_1,\ldots,I_q,M)$	HASH(x,j)
Return $(b' = b)$	If not $T[x,j]$ then $T[x,j] \twoheadleftarrow \{0,1\}^{\mathsf{H.ol}}$
HASH(x,j)	$Q \leftarrow Q \cup \{x\}$; Return $T[x,j]$
If not $T[x,j]$ then	
If $b = 0$ then $T[x,j] \twoheadleftarrow \{0,1\}^{\mathsf{H.ol}}$	
Else $T[x,j] \leftarrow \mathsf{H}(I_j,x)$	
Return $T[x,j]$	

Fig. 11. Games $\mathbf{G}^{\mathrm{uce}}$ and $\mathbf{G}^{\mathrm{sp}}$ to define UCE security.

H with $\mathsf{H.ol} = \kappa$ and $\mathsf{H.il} = p$. The selector is of length $r = \mathsf{H.kl} + p \cdot \lg(k)$ and specifies a key I for H as well as the probe sequence $\mathbf{p}$. The derived key K is then computed as shown. The following theorem says that the scheme works, meaning achieves our notion of encapsulation security. This involves two claims. First is that the key encapsulation advantage can be bounded by the uce advantage of a source-distinguisher pair. But this by itself is not enough. To ensure this uce advantage is small, we also show that the predictability of the source can be bounded. Here we appeal to our bound on sub-key predictability, so that once again the latter emerges as crucial.

$$\frac{\text{Algorithm } \mathsf{XKEY2}_{k,\kappa,p,r}(\boldsymbol{K}, R)}{(I, \mathbf{p}) \leftarrow R \ ; \ J \leftarrow \boldsymbol{K}[\mathbf{p}] \ ; \ K \leftarrow \mathsf{H}(I, J) \ ; \ \text{Return } K}$$

Fig. 12. Encapsulation algorithm $\mathsf{XKEY2}$. Given a length-k big-key $\boldsymbol{K}$ and a length-r selector $R = (I, \mathbf{p})$, the algorithm returns a length-κ subkey K.

Theorem 14. *Let $k, \kappa, p \geq 1$ be integers and H a family of functions with $\mathsf{H.ol} = \kappa$ and $\mathsf{H.il} = p$. Let $r = \mathsf{H.kl} + p \cdot \lg(k)$. Let $\mathsf{KEY} = \mathsf{XKEY2}_{k,\kappa,p,r}$ be the big-key key-derivation scheme associated to them as per Fig. 12. Let $\mathcal{A}$ be an adversary making at most q queries to its* DERIVE *oracle and leaking ℓ bits. The proof specifies a source adversary $\mathcal{S}$ and a distinguisher adversary $\mathcal{D}$ such that*

$$\mathsf{Adv}^{\mathrm{key}}_{\mathsf{KEY}}(\mathcal{A}) \ \leq \ \mathsf{Adv}^{\mathrm{uce}}_{\mathsf{H},q}(\mathcal{S}, \mathcal{D}) \quad \textit{and} \quad \mathsf{Adv}^{\mathrm{pred}}_{\mathcal{S}} \leq \mathsf{Adv}^{\mathrm{skp}}_{k,p,q}(\ell) \ . \tag{21}$$

Adversary $\mathcal{S}$ makes q queries to its HASH *oracle (one per key) and the running times of $\mathcal{S}$ and $\mathcal{D}$ add up to essentially that of $\mathcal{A}$.* ▌

Proof (Theorem 14). Adversaries $\mathcal{S}, \mathcal{D}$ are as follows:

$\underline{\text{Adversary } \mathcal{S}^{\text{HASH}}}$	$\underline{\text{Adversary } \mathcal{D}(I_1, \ldots, I_q, M)}$
$\boldsymbol{K} \twoheadleftarrow \{0,1\}^k \ ; \ (\mathsf{LK}, \sigma) \twoheadleftarrow \mathcal{A}(\,)$	$(L, \sigma, \mathbf{p}_1, \ldots, \mathbf{p}_q, K_1, \ldots, K_q) \leftarrow M$
$L \twoheadleftarrow \mathsf{LK}(\boldsymbol{K})$	$i \leftarrow 0 \ ; \ c' \twoheadleftarrow \mathcal{A}^{\text{DERIVE}}(L, \sigma)$
For $i = 1, \ldots, q$ do	Return c'
$\quad \mathbf{p}_i \twoheadleftarrow [1..k]^p \ ; \ J_i \leftarrow \boldsymbol{K}[\mathbf{p}_i]$	$\underline{\text{DERIVE}()}$
$\quad K_i \leftarrow \text{HASH}(J_i, i)$	$i \leftarrow i + 1 \ ; \ \text{return } ((I_i, \mathbf{p}_i), K_i)$
$M \leftarrow (L, \sigma, \mathbf{p}_1, \ldots, \mathbf{p}_q, K_1, \ldots, K_q)$	
Return M	

Adversary $\mathcal{S}$ itself picks the big-key $\boldsymbol{K}$ and runs $\mathcal{A}$ to get the leakage function, producing its own leakage M as shown. Adversary $\mathcal{D}$ continues the execution of $\mathcal{A}$, being in a position to answer DERIVE queries because it has $I_1, \ldots, I_q$. If $\mathcal{P}$ is any predictor adversary, the leakage M it gets in its game specifies the output L of the leakage function on the big-key, the probe sequences, and independent random strings $K_1, \ldots, K_q$, and an oracle query of the source is a subkey, so guessing it is exactly guessing a subkey. It follows that $\mathsf{Adv}^{\mathrm{pred}}_{\mathcal{S}}(\mathcal{P}) \leq \mathsf{Adv}^{\mathrm{skp}}_{k,p,q}(\ell)$. We omit the details. ▌

Big-key encryption. We can turn XKEY2 into a big-key encryption scheme via the general transform of Sect. 5. This transform does not introduce a random oracle. (If the big-key key encapsulation mechanism used one, it will inherit it, but will not introduce an additional use.) Thus the result of applying the transform to XKEY2 is a standard-model big-key encryption scheme. It satisfies locality because XKEY2 does. Theorem 13 reduces its security to that of XKEY2, and thus we can conclude by applying Theorem 14. We omit the details.

7 Authenticity and Hedged Big-Key Encryption

In real-world settings we are likely to want authenticated encryption (AE) rather than privacy-only encryption. We should thus ask whether, we can have big-key AE rather than the privacy-only formulation we have now. As discussed in Sect. 1, this is not possible due to the following attack: the adversary simply leaks a valid ciphertext. This is a small amount of leakage, yet violates authenticity.

To overcome this difficulty we suggest to use what we call *hedged* big-key encryption. This provides privacy in the big-key setting we have already defined and achieved; additionally, in the absence of leakage, it provides authenticity. We suggest that this is a good goal because, in the mass-surveillance /APT context, it is privacy that is the main concern, not authenticity; but in the absence of an APT, our concerns would be the usual ones, which include authenticity. Hedged big-key encryption provides both, so that security does not degrade by moving to big keys.

There is a simple and generic way to turn a privacy-only big-key encryption scheme into a hedged big-key encryption scheme. Reserve a small (128-bit, say) portion K of the big-key $\boldsymbol{K}$ as a key for a conventional PRF or MAC. Then use encrypt-then-mac [10]. Namely, big-key encrypt the message under the remaining (big) portion of $\boldsymbol{K}$ to get a ciphertext C, and return (C, T) as the ciphertext for the hedged big-key scheme, where T is the result of applying a PRF, keyed by K, to C. In the absence of leakage, we have authenticated encryption by applying results of [10]. In the presence of leakage, we must assume the small key K is leaked in its entirety, but the big-key privacy-only component will still provide the same privacy as before. Here we use the fact that in the privacy proof of [10], the adversary can be given the PRF (MAC) key.

Acknowledgments. Bellare was supported in part by NSF grants CNS-1526801 and CNS-1228890, a gift from Microsoft corporation and ERC Project ERCC (FP7/615074). Rogaway was supported in part by NSF grants CNS-1228828 and CNS-1314885. We thank Joseph Jaeger for comments and corrections, Wei Dai for helpful discussions, and the CRYPTO 2016 reviewers for their knowledgeable reviews, corrections and pointers to the literature.

References

1. Agrawal, D., Archambeault, B., Rao, J.R., Rohatgi, P.: The EM side-channels. In: Kaliski Jr., B.S., Koç, Ç.K., Paar, C. (eds.) CHES 2002. LNCS, vol. 2523. Springer, Heidelberg (2003)
2. Alwen, J., Dodis, Y., Naor, M., Segev, G., Walfish, S., Wichs, D.: Public-key encryption in the bounded-retrieval model. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 113–134. Springer, Heidelberg (2010)
3. Alwen, J., Dodis, Y., Wichs, D.: Leakage-resilient public-key cryptography in the bounded-retrieval model. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 36–54. Springer, Heidelberg (2009)
4. Alwen, J., Dodis, Y., Wichs, D.: Survey: leakage resilience and the bounded retrieval model. In: Kurosawa, K. (ed.) Information Theoretic Security. LNCS, vol. 5973, pp. 1–18. Springer, Heidelberg (2010)
5. Aumann, Y., Ding, Y.Z., Rabin, M.O.: Everlasting security in the bounded storage model. IEEE Trans. Inf. Theory **48**(6), 1668–1680 (2002)
6. Aumann, Y., Rabin, M.O.: Information theoretically secure communication in the limited storage space model. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 65–79. Springer, Heidelberg (1999)
7. Bellare, M., Desai, A., Jokipii, E., Rogaway, P.: A concrete security treatment of symmetric encryption. In: 38th FOCS, pp. 394–403. IEEE Computer Society Press, October 1997
8. Bellare, M., Hoang, V.T., Keelveedhi, S.: Instantiating random oracles via UCEs. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013, Part II. LNCS, vol. 8043, pp. 398–415. Springer, Heidelberg (2013)
9. Bellare, M., Kane, D., Rogaway, P.: Big-key symmetric encryption: resisting key exfiltration. Cryptology ePrint Archive, report 2016/541 (2016)
10. Bellare, M., Namprempre, C.: Authenticated encryption: relations among notions and analysis of the generic composition paradigm. In: Okamoto, T. (ed.) ASIACRYPT 2000. LNCS, vol. 1976, pp. 531–545. Springer, Heidelberg (2000)
11. Bellare, M., Paterson, K.G., Rogaway, P.: Security of symmetric encryption against mass Surveillance. In: Garay, J.A., Gennaro, R. (eds.) CRYPTO 2014, Part I. LNCS, vol. 8616, pp. 1–19. Springer, Heidelberg (2014)
12. Bellare, M., Rogaway, P.: Random oracles are practical: a paradigm for designing efficient protocols. In: Ashby, V. (ed.) ACM CCS 1993, pp. 62–73. ACM Press, November 1993
13. Bellare, M., Rogaway, P.: The Security of triple encryption and a framework for code-based game-playing proofs. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 409–426. Springer, Heidelberg (2006)
14. Boyen, X.: Reusable cryptographic fuzzy extractors. In: Atluri, V., Pfitzmann, B., McDaniel, P. (eds.) ACM CCS 2004, pp. 82–91. ACM Press, October 2004
15. Brzuska, C., Farshim, P., Mittelbach, A.: Indistinguishability obfuscation and UCEs: the case of computationally unpredictable sources. In: Garay, J.A., Gennaro, R. (eds.) CRYPTO 2014, Part I. LNCS, vol. 8616, pp. 188–205. Springer, Heidelberg (2014)
16. Cachin, C., Maurer, U.M.: Unconditional security against memory-bounded adversaries. In: Kaliski Jr., B.S. (ed.) CRYPTO 1997. LNCS, vol. 1294, pp. 292–306. Springer, Heidelberg (1997)
17. Cash, D.M., Ding, Y.Z., Dodis, Y., Lee, W., Lipton, R.J., Walfish, S.: Intrusion-resilient key exchange in the bounded retrieval model. In: Vadhan, S.P. (ed.) TCC 2007. LNCS, vol. 4392, pp. 479–498. Springer, Heidelberg (2007)

18. Cramer, R., Shoup, V.: Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack. SIAM J. Comput. **33**(1), 167–226 (2003)
19. Dagon, D., Lee, W., Lipton, R.J.: Protecting secret data from insider attacks. In: Patrick, A.S., Yung, M. (eds.) FC 2005. LNCS, vol. 3570, pp. 16–30. Springer, Heidelberg (2005)
20. Di Crescenzo, G., Lipton, R.J., Walfish, S.: Perfectly secure password protocols in the bounded retrieval model. In: Halevi, S., Rabin, T. (eds.) TCC 2006. LNCS, vol. 3876, pp. 225–244. Springer, Heidelberg (2006)
21. Dodis, Y., Kalai, Y.T., Lovett, S.: On cryptography with auxiliary input. In: Mitzenmacher, M. (ed.) 41st ACM STOC, pp. 621–630. ACM Press, May/June 2009
22. Dodis, Y., Ristenpart, T., Vadhan, S.: Randomness condensers for efficiently samplable, seed-dependent sources. In: Cramer, R. (ed.) TCC 2012. LNCS, vol. 7194, pp. 618–635. Springer, Heidelberg (2012)
23. Dziembowski, S.: Intrusion-resilience via the bounded-storage model. In: Halevi, S., Rabin, T. (eds.) TCC 2006. LNCS, vol. 3876, pp. 207–224. Springer, Heidelberg (2006)
24. Dziembowski, S., Maurer, U.M.: Optimal randomizer efficiency in the bounded-storage model. J. Cryptol. **17**(1), 5–26 (2004)
25. Dziembowski, S., Pietrzak, K.: Leakage-resilient cryptography. In: 49th FOCS, pp. 293–302. IEEE Computer Society Press, October 2008
26. Halevi, S., Lin, H.: After-the-fact leakage in public-key encryption. In: Ishai, Y. (ed.) TCC 2011. LNCS, vol. 6597, pp. 107–124. Springer, Heidelberg (2011)
27. Kelsey, J., Schneier, B.: Authenticating secure tokens using slow memory access. In: Proceedings of the USENIX Workshop on Smartcard Technology (Smartcard 1999), 10–11 May 1999, Chicago, Illinois, USA, p. 101. USENIX Association (1999)
28. Lu, C.-J.: Hyper-encryption against space-bounded adversaries from on-line strong extractors. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 257–271. Springer, Heidelberg (2002)
29. Maurer, U.M.: Conditionally-perfect secrecy and a provably-secure randomized cipher. J. Cryptol. **5**(1), 53–66 (1992)
30. Nisan, N., Zuckerman, D.: Randomness is linear in space. J. Comput. Syst. Sci. **52**(1), 43–52 (1996)
31. Pietrzak, K.: A leakage-resilient mode of operation. In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 462–482. Springer, Heidelberg (2009)
32. Raz, R., Reingold, O.: On recycling the randomness of states in space bounded computation. In: 31st ACM STOC, pp. 159–168. ACM Press, May 1999
33. Reingold, O., Shaltiel, R., Wigderson, A.: Extracting randomness via repeated condensing. SIAM J. Comput. **35**(5), 1185–1209 (2006)
34. Shin, S.H., Kobara, K., Imai, H.: Leakage-resilient authenticated key establishment protocols. In: Laih, C.-S. (ed.) ASIACRYPT 2003. LNCS, vol. 2894, pp. 155–172. Springer, Heidelberg (2003)
35. Vadhan, S.P.: Constructing locally computable extractors and cryptosystems in the bounded-storage model. J. Cryptol. **17**(1), 43–77 (2004)