# Defending Against Key Exfiltration: Efficiency Improvements for Big-Key Cryptography via Large-Alphabet Subkey Prediction

Mihir Bellare*
Department of Computer Science and Engineering
University of California, San Diego
mihir@eng.ucsd.edu

Wei Dai†
Department of Computer Science and Engineering
University of California, San Diego
weidai@eng.ucsd.edu

## ABSTRACT

Towards advancing the use of big keys as a practical defense against key exfiltration, this paper provides efficiency improvements for cryptographic schemes in the bounded retrieval model (BRM). We identify probe complexity (the number of scheme accesses to the slow storage medium storing the big key) as the dominant cost. Our main technical contribution is what we call the large-alphabet subkey prediction lemma. It gives good bounds on the predictability under leakage of a random sequence of blocks of the big key, as a function of the block size. We use it to significantly reduce the probe complexity required to attain a given level of security. Together with other techniques, this yields security-preserving performance improvements for BRM symmetric encryption schemes and BRM public-key identification schemes.

## 1 INTRODUCTION

This paper is concerned with the threat of key exfiltration. This means attacker-planted malware on the key-storing system uses the system's network connection to convey the key to a remote accomplice. A line of theoretical work has suggested a mitigation, called the Bounded Retrieval Model (BRM) [1, 2, 10, 13, 14], which involves using big keys. BKR [5] initiated an effort to take the BRM (they call it big-key cryptography) to practicality. We continue this effort. We identify probe complexity (the number of scheme accesses to the slow storage medium storing the big key) as the dominant cost. Our *large-alphabet subkey prediction lemma* allows us to minimize the probe complexity required to reach a given level of security, thereby optimizing storage usage. We use this to obtain efficiency improvements for big-key symmetric encryption [5]. We then provide an additional lemma on polynomial-evaluation entropy preservation, and use the two lemmas in conjunction to obtain efficiency improvements for the ADW big-key identification scheme [2].

*Supported in part by NSF grants CNS-1526801 and CNS-1717640, ERC Project ERCC FP7/615074 and a gift from Microsoft corporation.
†Supported in part by a Powell Fellowship and grants of first author.


*CCS '17, October 30-November 3, 2017, Dallas, TX, USA*

ACM ISBN 978-1-4503-4946-8/17/10…$15.00
https://doi.org/10.1145/3133956.3133965

*Large-alphabet subkey prediction.* Let $b \geq 2$ be an integer representing the *block size* in a storage system, for example $b = 32$ or $b = 64$ for words in memory, or $b = 8 \cdot 512$ (512 bytes) for a typical hard-disk drive. Let $q = 2^b$ be the *alphabet size*, and $[q] = \{0, 1, \ldots, q-1\}$ the corresponding alphabet. Let $\boldsymbol{K} = (\boldsymbol{K}[0], \ldots, \boldsymbol{K}[k-1]) \in [q]^k$ be a string over $[q]$ of length $k$, randomly chosen. It represents a (big) key stored in our storage system as a sequence of $k$ blocks. We imagine that an adversary-chosen function $\mathsf{Lk}$: $[q]^k \to [q]^\ell$ (representing adversary-implanted malware, and here called the leakage function) is applied to $\boldsymbol{K}$, and the result $L$ (representing exfiltrated information, here called the leakage), is provided back to the adversary. Think of $\ell$ as somewhat smaller than $k$, for example $\ell \leq k/10$, so that the leakage, although not total, is certainly non-trivial. Despite this, we wish to make secure use of the big key, specifically to (repeatedly) extract "small" keys ($\tau \geq 1$ blocks, for a parameter $\tau$) for use with conventional cryptography. In any such extraction, we make $\tau$ random but distinct probes $i_1, \ldots, i_\tau \in [k] = \{0, 1, \ldots, k-1\}$ into $\boldsymbol{K}$ to determine $J = \boldsymbol{K}[i_1] \ldots \boldsymbol{K}[i_\tau]$ as the $\tau$-block short key. Given the leakage $L$ and the probe positions $i_1, \ldots, i_\tau$, the adversary aims to predict (compute in its entirety) $J$. Two metrics (see Section 3 for precise definitions of what we discuss next) are of interest. First is the *subkey prediction advantage*

$$\mathbf{Adv}^{\mathrm{skp}}_{q,k,\tau}(\ell)\,, \tag{1}$$

defined as the maximum probability that an adversary can compute $J$, the maximum being over all leakage functions $\mathsf{Lk}$ returning $\ell$ blocks and over all adversary strategies. It is useful to let $k^* = kb$ denote the amount of storage occupied by the big key in bits, and, correspondingly, $\ell^* = \ell b$ the amount of allowed leakage in bits. (We will want to fix these and vary $b$, thereby defining $k$ and $\ell$.) Now, in usage, we would ask for a certain number $s$ of bits of security (for example $s = 128$), meaning we want the subkey prediction advantage to be at most $2^{-s}$, and then want to know the number $\tau$ of probes it takes to get there. This is the *probe complexity*,

$$\mathbf{Probes}_{k^*,\ell^*,s}(b) = \min\left\{\tau \,:\, \mathbf{Adv}^{\mathrm{skp}}_{2^b,k^*/b,\tau}(\ell^*/b) \leq 2^{-s}\right\}\,. \tag{2}$$

The probe complexity will be our cost in accesses to a potentially slow storage system, like a disk, and for effiency of the overlying big-key scheme, we want to minimize it. To this end, Theorem 3.1 gives a good upper bound on the subkey prediction advantage, whence we obtain a good upper bound on the probe complexity. Next, we compare our bounds to prior ones, and discuss history and applications (to big-key cryptography and thus key exfiltration resistance).

| $b$ | $s = 128$ | |
|---|---|---|
| | Us | ADW |
| 1 | 271 | 11532 |
| 8 | 61 | 1584 |
| 32 | 47 | 592 |
| 64 | 45 | 434 |
| $8 \cdot 512$ | 43 | 287 |
| $8 \cdot 4096$ | 43 | 285 |

| $b$ | $s = 512$ | |
|---|---|---|
| | Us | ADW |
| 1 | 971 | 46127 |
| 8 | 219 | 6335 |
| 32 | 171 | 2366 |
| 64 | 165 | 1735 |
| $8 \cdot 512$ | 159 | 1146 |
| $8 \cdot 4096$ | 158 | 1139 |

**Figure 1: Fix the amount of storage we allocate to the big key at $k^* = 8 \cdot 10^{11}$ bits = 100 GBytes. Fix the amount of leakage at 10% of the length of the big key, $\ell^* = k^*/10$ = 10 GBytes. The first table considers security level $s = 128$, while the second considers $s = 512$. Each table then considers different block sizes $b$. (Once $b$ is chosen, the length of the big key in blocks is $k = k^*/b$ and the length of the leakage in blocks is $\ell = \ell^*/b$.) The table entries show upper bounds on the probe complexity $\mathbf{Probes}_{k^*,\ell^*,s}(b)$. The "Us" column is our bound via Theorem 3.1, and "ADW" is what is obtained via [3, Lemma A.3]. The block sizes are chosen to represent common word or disk sector sizes in storage systems.**

*Prior work and comparisons.* ADW [3, Lemma A.3] is an elegant and general result that, as a special case, gives an upper bound on the subkey prediction advantage (and thus probe complexity) for all values of parameters we consider. The bounds, however are quite poor, so that, to get a desired level of security, one needs a very large number of probes (we will see numbers in a bit), resulting in a significant loss of efficiency for the overlying big-key cryptography schemes. This lead BKR [5] (in their quest for practical big-key symmetric encryption) to formulate subkey prediction, and seek better bounds by direct analysis. They however *only considered the case* $b = 1$ of a binary alphabet. They gave an example to show that there are non-obvious leakage functions that lead to better subkey prediction advantage than one might expect, making the problem of giving a (good) upper bound challenging. Via a combinatorial analysis, they showed that the worst case is when the pre-images of the outputs of the leakage function are approximate Hamming balls in the space of big keys, thereby deriving estimates (not quite upper bounds, something we rectify) on the subkey prediction advantage and probe complexity, for the case $b = 1$ ($q = 2$), that are much better than those obtained via ADW [2, Lemma A.3]. They posed the large alphabet ($b > 1$) case as an open question, asking, specifically, to give bounds on subkey prediction advantage and probe complexity, in the $b > 1$ case, that are better than the ones obtained via ADW [3, Lemma A.3]. (The motivation, as we will see later, was to improve efficiency of big-key symmetric encryption.) Our work answers this question, giving (good) upper bounds as a function of the block size $b$.

In usage, we would typically first decide on the amount of storage $k^*$ (measured in bits) we allocate to the big key, for example $k^* = 8 \cdot 10^{11}$ bits = 100 GBytes. Next we would fix the amount of leakage $\ell^*$ (also measured in bits), for example $\ell^* = k^*/10 = 10$ GBytes, corresponding to 10% of the length of the big key. The block size $b$ may be determined by the storage system (for example 512 bytes or 4096 bytes) or chosen to optimize security and efficiency as per our bounds. Once it is chosen, the length in blocks $k = k^*/b$ of the big key and $\ell = \ell^*/b$ of the leakage are determined. Now, for a given level $s$ of security, we want to know the probe complexity $\mathbf{Probes}_{k^*,\ell^*,s}(b)$. Smaller (fewer probes into the likely slow storage system) is better. We tabulate results in Fig. 1. Our bounds emerge as substantially better than those obtained via ADW [3, Lemma A.3]. For example, for $s = 128$, the improvement ranges from a factor of 26 ($b = 8$) to a factor of 6.6 ($b = 8 \cdot 4096$). Below, we will see how this translates to efficiency improvements for big-key cryptography.

*The BRM.* Assume (for concreteness of this discussion) that the primitive is symmetric encryption [5] (we will discuss other primitives later), and let $\boldsymbol{K}$ denote the encryption key, $k^*$ bits long. In the Bounded Retrieval Model (BRM) [1, 2, 5, 10, 13, 14], an adversary-chosen function Lk (representing adversary-implanted malware) is applied to $\boldsymbol{K}$, and the $\ell^*$-bit result $L$ (representing the exfiltrated information), is provided back to the adversary. Security would appear impossible, since Lk could be the identity function, so that $L = \boldsymbol{K}$, but the idea is that $\boldsymbol{K}$ is big (for example $k^* = 100$ GBytes), while $L$ is assumed to be somewhat smaller (like $\ell^* = k^*/10 = 10$ GBytes). In other words, the model assumes that the amount of data exfiltrated can be limited, say via network or system monitoring. Indeed, security product vendors such as McAfee [16] provide tools for this type of monitoring and detection.

If the scheme is poorly designed, the fact that the exfiltrated information is somewhat shorter than the key won't guarantee security. For example if the scheme applies SHA256 to $\boldsymbol{K}$ to get a 256 bit key $K$ and then uses AES256 to encrypt the data, then $\mathsf{Lk}(\boldsymbol{K})$ can just return the 256 bit string $K = \mathsf{SHA256}(\boldsymbol{K})$ and security is entirely compromised no matter how big is $\boldsymbol{K}$. The first requirement for a BRM (also called big-key) scheme is thus *leakage resilience*, meaning an adversary, given $L = \mathsf{Lk}(\boldsymbol{K})$, still cannot violate security, and this must be true for *any* (adversary-chosen) function Lk that returns $\ell^*$ bits.

*Probe complexity.* Big keys may help for security, but it would be prohibitively costly to process a 100 GByte key for every encryption. The BRM addresses this via a condition that says that each encryption (or decryption) operation should only make a "small" number of probes into the big key $\boldsymbol{K}$, meaning have low probe complexity. Security in the presence of leakage is a difficult goal under any circumstances, but made even more so here by this requirement.

*From bits to blocks.* Viewing the big key $\boldsymbol{K} = (\boldsymbol{K}[0], \ldots, \boldsymbol{K}[k^*-1])$ as a sequence of bits, BKR encryption [5] begins by making some $\tau^*$ random probes $i_1, \ldots, i_{\tau^*} \in [k^*]$ into $\boldsymbol{K}$ to extract a $\tau^*$-bit subkey $J = \boldsymbol{K}[i_1] \ldots \boldsymbol{K}[i_{\tau^*}]$. It then applies a (randomized) hash function to $J$ to get a key $K$ for a conventional (AES-based) symmetric encryption scheme, and uses $K$ to encrypt the data. Once $J$ has been obtained, the computation, being symmetric cryptography operations, is quite efficient, but $\boldsymbol{K}$, being big, is likely stored on a slow medium like a hard drive, and so the encryption cost is dominated by the storage accesses needed to get $J$. For a subkey prediction advantage of $s = 128$ (based on which BKR show ind-cpa style security of their encryption scheme at the same security level), BKR will need $\tau^* = 271$ probes into the storage. (This is as per the $b = 1$ row of the first

table in Fig. 1. BKR's subkey prediction lemma gives an estimate, not a bound, so we use our number, but numerically the two are almost the same.)

But (as BKR themselves point out), their scheme is making very poor use of storage by drawing only a bit of the big key per probe. Letting $b$ be some appropriate block size determined by the storage system (for example $b = 8 \cdot 512$ bits = 512 Bytes), $\boldsymbol{K}$ would actually be stored as a sequence of blocks, and a single probe into the storage can retrieve an entire block at about the same cost as retrieving a single bit. Indeed, a typical storage API does not even provide a way to directly access a bit, so an implementation of BKR would, for a probe for bit-position $j$, have to draw the block containing this bit position, take the corresponding bit, and throw the other bits away. A natural improvement (suggested by BKR) is to draw (and use) an entire block per probe. Thus, we now view the big key $\boldsymbol{K} = (\boldsymbol{K}[0], \ldots, \boldsymbol{K}[k-1]) \in [2^b]^k$ as a sequence of blocks, corresponding to the way it is actually stored, where $k = k^*/b$ is the number of blocks. Now, making some $\tau$ probes $i_1, \ldots, i_\tau \in [k]$ into $\boldsymbol{K}$, one obtains the subkey $J = \boldsymbol{K}[i_1] \ldots \boldsymbol{K}[i_\tau]$. The rest of the encryption process is as before, and as we have already noted, is efficient, even though $J$ could be a bit longer. Continuing to require a subkey prediction advantage of $s = 128$, the question is, what value of $\tau$ guarantees this? This is the question that BKR could not answer. It is answered by our large-alphabet subkey prediction lemma. Specifically, the first table of Fig. 1 gives values of $\tau$ for different choices of $b$. For $b =$ 512 Bytes, we see that $\tau = 43$. Recalling that BKR needed $\tau^* = 271$ probes, we have reduced the number of probes (storage accesses) by a factor of $271/43 \approx 6$, meaning offer a 6x speedup.

The price we pay (as alluded to above) is that $J$ is longer, specifically, 271 bits for BKR and $43 \cdot 512 \approx 22$ KBytes for us. This means the hashing of $J$ to obtain the AES key $K$ takes longer, but modern hash functions are fast, and the time saved in storage accesses is more than the time lost in extra hashing [11, 12]. This is especially true since the hashing can be pipelined, taking advantage of the iterated structure of hash functions to process blocks incrementally as soon as they are retrieved rather than delaying hashing until after all blocks are retrieved.

*Big-key identification.* In a (public-key) identification scheme, a user (called the prover) has a secret key sk whose associated public key pk is held by the server (called the verifier). Access control is enforced by having the prover identify itself as the owner of sk via an interactive identification protocol. The Schnorr [19] and Okamoto [17] schemes are well-known examples, but they are of course conventional (small-key) schemes. Identification is an interesting target for a BRM scheme. Here it is the secret key sk that would be big (100 GBytes)— we want the public key pk to remain of conventional size. The usage we envision is hardware-assisted access control, where sk is on an auxiliary device like a USB stick that the user plugs into a possibly infected machine to identify herself (login) to the server across the network. The key being large, and reading from a USB being slow, the malware will have difficulty obtaining enough information about the key (10 GBytes) to violate BRM security, even after a significant number of usages (logins) by the user.

Identification in the BRM was first treated by ADW [2], who gave (asymptotic) security definitions and a clever scheme that involves combining multiple instances of the Okamoto scheme [17] in a compact way. We target making this scheme practical. The quest is meaningless in the absence of concrete security, for practicality is fundamentally about maximizing efficiency for a given level (eg. 128 bits) of security. A first and central step is thus a *concrete-security treatment.* We render the definitions of big-key identification (the goal is security against impersonation under active attack) concretely, then revisit the asymptotically-stated result of ADW [2] to render it, too, in concrete form. We note that for the ADW scheme, probe complexity dictates the computational cost of the two most costly phases of the protocol, the response phase and verification phase (as we will demonstrate in Fig. 11). Hence, improvements in probe complexity directly translate into improvements in efficiency. Towards lowering probe complexity for a given level of security, we first *improve the concrete security* of the reduction via a lemma on the entropy preservation of polynomial evaluation that improves bounds from ADW [2]. We then obtain further reductions in probe complexity, by *using our large-alphabet subkey prediction lemma* in place of ADW's own [3, Lemma A.3]. The large-alphabet aspect here is crucial, for the scheme draws, from the big key, a value in $\mathbb{Z}_p^m$, where $p$ is a prime of 512 bits long (for 128-bit security of the identification scheme), and $m \geq 2$ is an integer parameter, so probes need to return blocks of the (large) size $b = m \cdot \lceil \log_2(p) \rceil$. Putting it all together gives a reasonable-cost big-key identification scheme, and the first concrete rendition of the ADW big-key identification scheme. A preliminary implementation shows that with a pairing-friendly group of 512 bits, the execution of the protocol takes a few seconds.

## 2 PRELIMINARIES

For $n$ a positive integer, we let $[n] = \{0, 1, \ldots, n-1\}$, and $[1..n] = \{1, \ldots, n\}$. We also use the notation $\mathbb{Z}_n$ to denote the set $[n]$ in contexts where we use the underlying algebraic structure modulo $n$. If $\mathbf{x}$ is a vector, then $|\mathbf{x}|$ denotes its length and $\mathbf{x}[i]$ denotes its $i$-th coordinate. We call $\mathbf{x}$ an $n$-vector if $|\mathbf{x}| = n$. We number coordinates starting from 0. For example if $\mathbf{x} = (10, 0, 11)$ then $|\mathbf{x}| = 3$ and $\mathbf{x}[2] = 11$. We let $\varepsilon$ denote the empty vector, which has length 0. If $0 \leq i \leq |\mathbf{x}| - 1$ then we let $\mathbf{x}[0..i] = (\mathbf{x}[0], \ldots, \mathbf{x}[i])$, this being $\varepsilon$ when $i = 0$. We say that $\mathbf{x}$ is a vector over set $S$ if all its coordinates belong to $S$. We let $S^n$ denote the set of all $n$-vectors over $S$ and we let $S^*$ denote the set of all finite-length vectors over the set $S$. If $S$ is a set then $|S|$ denotes its size. If $\tau \leq |S|$ is a positive integer, we let $S^{(\tau)}$ be the set of $\tau$-vectors over $S$ with distinct entries. 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, $x[0..i] = x[0] \ldots x[i]$, $\varepsilon$ is the empty string, $\{0, 1\}^n$ is the set of $n$-bit strings and $\{0, 1\}^*$ the set of all strings. For $\mathbf{K} \in [q]^k$ and $\mathbf{p} \in [k]^*$, we let $\mathbf{K}[\mathbf{p}] = (\mathbf{K}[\mathbf{p}[0]], \mathbf{K}[\mathbf{p}[1]], \ldots, \mathbf{K}[\mathbf{p}[|\mathbf{p}| - 1]])$, this being $\varepsilon$ when $\mathbf{p} = \varepsilon$.

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 use the code-based game-playing framework [8] (see Fig. 2 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.

Following [7], our random oracle H is variable range. This means it takes two inputs, $x$ and Rng, where the latter is a (description of) an efficiently sampleable set, and returns as output a random point in Rng. In schemes, we (implicitly or explicitly) fix a unique description for each range set that is used. For example, $x \xleftarrow{\$} \mathsf{H}(x, [k])$ will return a random element in $[k]$, and $\mathbf{p} \xleftarrow{\$} \mathsf{H}(x, [k]^{(\tau)})$ will return a random $\tau$-dimensional vector over $[k]$ *with distinct entries.*

## 3 LARGE-ALPHABET SUBKEY PREDICTION

Here we define the subkey prediction problem parameterized by alphabet size and give our results about it.

### 3.1 The Problem

We consider the subkey-prediction game, $\mathbf{G}^{\mathrm{skp}}_{q,k,\tau}(\mathcal{A}, \mathsf{Lk})$, shown on the left of Fig. 2. (Ignore the other game for now.) The quantities involved in the game, as well as associated ones, are as follows:

- $q \geq 2$— the alphabet size. A *block* is an element of $[q]$.
- $b \geq 1$— the block length, meaning $q = 2^b$. Theorem 3.1 does not assume $q$ is a power of two, but it is in some applications.
- $k$— the length in blocks of the big key
- $\mathsf{Lk}$: $[q]^k \to [q]^\ell$— the leakage function
- $\ell$— the length of the output of the leakage function, called the leakage length, in blocks
- $L$— the leakage, an $\ell$-vector over $[q]$ returned by $\mathsf{Lk}$
- $\boldsymbol{K}$— the big key, a vector of length $k$ over $[q]$
- $\tau \leq k$— the number probes into the big key $\boldsymbol{K}$
- $\mathbf{p}$— the probe vector, a $\tau$-vector over $[k]$ all of whose coordinates are distinct
- $\mathcal{A}$— the adversary
- $k^*$— the length of the big key in bits, $k^* = kb$
- $\ell^*$— the length of the leakage in bits, $\ell^* = \ell b$
- $\rho = \ell^*/k^* = \ell/k$— the leakage rate.

In the game on the left in Fig. 2, a $k$-vector $\boldsymbol{K}$ over $[q]$, called the big key, is randomly chosen from $[q]^k$. Then, a random $\tau$-vector $\mathbf{p}$ is chosen from $[k]^{(\tau)}$, so that its coordinates are all distinct. (Recall that $[k]^{(\tau)}$, the set from which $\mathbf{p}$ is selected in the game in Fig. 2, denotes the set of all $\tau$-vectors over $[k]$ all of whose coordinates are distinct.) We refer to $\mathbf{p}$ as the probe vector. Each of its coordinates is a probe, pointing to a location in the big key. Adversary $\mathcal{A}$ is given the leakage $L = \mathsf{Lk}(\boldsymbol{K})$ and the probe vector $\mathbf{p}$. Its goal is to predict (compute, output) $\boldsymbol{K}[\mathbf{p}] = (\boldsymbol{K}[\mathbf{p}[1]], \ldots, \boldsymbol{K}[\mathbf{p}[\tau]])$, the $\tau$-vector consisting of the coordinates of $\boldsymbol{K}$ selected by the coordinates of the probe vector. The adversary returns $J$ as its guess, and the game returns true if $\mathcal{A}$ succeeds, meaning $J = \boldsymbol{K}[\mathbf{p}]$. We define the following advantage metrics:

| **Game** $\mathbf{G}^{\mathrm{skp}}_{q,k,\tau}(\mathcal{A}, \mathsf{Lk})$ | **Game** $\mathbf{G}^{\mathrm{rskp}}_{k,\tau}(\mathcal{A}, \mathcal{K})$ |
|---|---|
| $\boldsymbol{K} \xleftarrow{\$} [q]^k$; $L \leftarrow \mathsf{Lk}(\boldsymbol{K})$ | $\boldsymbol{K} \xleftarrow{\$} \mathcal{K}$ |
| $\mathbf{p} \xleftarrow{\$} [k]^{(\tau)}$ | $\mathbf{p} \xleftarrow{\$} [k]^{(\tau)}$ |
| $J \xleftarrow{\$} \mathcal{A}(L, \mathbf{p})$ | $J \xleftarrow{\$} \mathcal{A}(\mathbf{p})$ |
| Return $(J = \boldsymbol{K}[\mathbf{p}])$ | Return $(J = \boldsymbol{K}[\mathbf{p}])$ |

**Figure 2: Left: Subkey prediction game $\mathbf{G}^{\mathrm{skp}}_{q,k,\tau}$. Right: Restricted subkey prediction game $\mathbf{G}^{\mathrm{rskp}}_{k,\tau}$ (used in Section 3.3).**

$$\mathbf{Adv}^{\mathrm{skp}}_{q,k,\tau}(\mathcal{A}, \mathsf{Lk}) = \Pr\left[\mathbf{G}^{\mathrm{skp}}_{q,k,\tau}(\mathcal{A}, \mathsf{Lk})\right],$$
$$\mathbf{Adv}^{\mathrm{skp}}_{q,k,\tau}(\mathsf{Lk}) = \max_{\mathcal{A}} \mathbf{Adv}^{\mathrm{skp}}_{q,k,\tau}(\mathcal{A}, \mathsf{Lk}),$$
$$\mathbf{Adv}^{\mathrm{skp}}_{q,k,\tau}(\ell) = \max_{\mathsf{Lk}:[q]^k \to [q]^\ell} \mathbf{Adv}^{\mathrm{skp}}_{q,k,\tau}(\mathsf{Lk}).$$

The first advantage is the probability that the game outputs true, meaning the probability that the adversary successfully returns $\boldsymbol{K}[\mathbf{p}]$. The second advantage is obtained by maximizing the first one over all adversaries $\mathcal{A}$. Note that this is well-defined since here we consider all computationally unbounded adversaries. The third advantage is obtained by maximizing the second advantage over all leakage functions $\mathsf{Lk}$ that output $\ell$ blocks.

Now fix some big-key length $k^*$ (in bits) and leakage length $\ell^*$ (in bits). Also fix an integer $s$ representing the desired security. For any block length $b \geq 1$ such that $b$ divides $k^*$ and $\ell^*$, we let

$$\mathbf{Probes}_{k^*,\ell^*,s}(b) = \min\left\{ \tau \,:\, \mathbf{Adv}^{\mathrm{skp}}_{2^b,k^*/b,\tau}(\ell^*/b) \leq 2^{-s} \right\}. \quad (3)$$

Here, we have set the alphabet size to $q = 2^b$. The length $k$ of the big key and $\ell$ of the leakage in blocks are determined, respectively, by $k = k^*/b$ and $\ell = \ell^*/b$. Then, $\mathbf{Probes}_{k^*,\ell^*,s}(b)$ is the smallest number of probes $\tau$ that will guarantee that $\mathbf{Adv}^{\mathrm{skp}}_{q,k,\tau}(\ell)$ is at most $2^{-s}$.

The subkey prediction game and problem formulated by BKR [5] differs in two ways. First, they had only considered the $q = 2$ case (that is, $b = 1$) of a binary alphabet. The large alphabet aspect of our treatment refers to the fact that our alphabet size is a parameter $q$ that we view as quite large. In some applications, $q = 2^b$ where $b$ is the block size of our storage medium, but Theorem 3.1 does not assume $q$ is a power of two. The second difference with BKR [5] is that their probes $\mathbf{p}[1], \ldots, \mathbf{p}[\tau]$ were random and independent, so in particular two of them might be the same, but ours are random subject to being distinct. This is important towards our being able to get a provable upper bound on the subkey prediction advantage, whereas BKR were only able to get (for their setting) an estimate or approximate upper bound.

Now our goal is to upper bound, as well as possible, the subkey prediction advantage $\mathbf{Adv}^{\mathrm{skp}}_{q,k,\tau}(\ell)$ as a function of $q, k, \tau, \ell$. Thence we will obtain upper bounds on $\mathbf{Probes}_{k^*,\ell^*,s}(b)$.

### 3.2 Subkey Prediction Theorem

The bound in our subkey prediction theorem is the ratio of the sizes of two $q$-ary hamming balls. We start with the relevant definitions.

*Hamming balls.* We define the weight of a $n$-vector $v$ over $[q]$ to be

$$w(v) = \left|\{i \in [n] \mid v[i] \neq 0\}\right| ,$$

the number of coordinates of $v$ that are non-zero. Let $\mathcal{K} \subseteq [q]^n$ for some integer $n$, we define the weight of $\mathcal{K}$ to be

$$w(\mathcal{K}) = \sum_{x \in \mathcal{K}} w(x) ,$$

the sum of weights of vectors in $\mathcal{K}$. For $0 \leq r \leq k$, the $q$-ary hamming ball of radius $r$ over $[q]^k$ is the set

$$\mathbf{B}_{q,k}(r) = \left\{ v \in [q]^k \ : \ w(v) \leq r \right\}$$

of $k$-vectors over $[q]$ that have more at most $r$ non-zero coordinates. We let $B_{q,k}(r)$ denote the size of the set $\mathbf{B}_{q,k}(r)$ and note that

$$B_{q,k}(r) = \sum_{i=0}^{r} (q-1)^i \binom{k}{i} .$$

For convenience of stating our results, we establish the following conventions: if $r > k$ then we let $B_{q,k}(r) = B_{q,k}(k) = q^k$, and if $k = 0$ then for all $r \geq 0$ we let $B_{q,k}(r) = 1$. We also define the function

$$\mathrm{rd}_{q,k}(N) = \max\left\{ r \in [k+1] \ : \ B_{q,k}(r) \leq N \right\}$$

to return the largest radius $r$ in the range $0 \leq r \leq k$ such that the ball $\mathbf{B}_{q,k}(r)$ has size at most $N$.

*Large-alphabet subkey prediction theorem.* We now give the main result of this section.

Theorem 3.1 (Subkey-prediction bound). *Let $q, k, \ell, \tau$ be integers with $q \geq 2$ and $\ell, \tau \leq k$. Let $r$ be any integer in the range $0 \leq r \leq \mathrm{rd}_{q,k}(q^{k-\ell})$. Then*

$$\mathbf{Adv}^{\mathrm{skp}}_{q,k,\tau}(\ell) \leq \frac{B_{q,k-\tau}(r)}{B_{q,k}(r)} . \tag{4}$$

The theorem allows us to pick the parameter $r$ arbitrarily in the given range, so for the best estimates we would pick a $r$ that minimizes the ratio. We postpone the proof to first discuss how this compares to prior work and how to use it to get numerical bounds.

*Comparison.* BKR [5] give an upper bound we denote $G^{\mathrm{bkr}}_{k,\tau}(2^{k-\ell})$ on the subkey prediction advantage in their setting. Recall that their setting differs from ours in two ways. First, $q = 2$ in their case. Second, in their game, the $\tau$ probes are random and independent, while in our game they are random but distinct. Their function $G^{\mathrm{bkr}}_{k,\tau}(N)$ is a sum of $\mathrm{rd}_{2,k}(N)$ terms. It is quite complex and it is hard to estimate numerically. BKR gave a simpler expression, that approximates $G^{\mathrm{bkr}}_{k,\tau}(N)$, and that they use for numerical estimates, but this expression is not an upper bound, and thus it is not clear their numerical estimates are upper bounds either. Our bound, the ratio of the sizes of two $q$-ary Hamming balls, is simpler than the bound of BKR (this makes crucial use of the probes being distinct), and, we will see, more analytically tractable, even when $q = 2$. In particular, we are able to upper bound $\min_r B_{q,k-\tau}(r)/B_{q,k}(r)$, subjected to $0 \leq r \leq \mathrm{rd}_{q,k}(q^{k-\ell})$, quite nicely for numerical estimates, as discussed next.

*Tools for deriving numerical bounds.* Theorem 3.1 upper bounds the subkey prediction advantage as the ratio of the sizes of two hamming balls. Below, we present tools to bound this ratio. First, we need some definitions. Let $H_2$ be the binary entropy function, defined for $x \in [0,1]$ by $H_2(x) = -x\log_2(x) - (1-x)\log_2(1-x)$. We note that the value of $x\log_q(x)$ is taken to be 0 when $x = 0$. This ensures that $H_2$ is continuous over $[0,1]$. More generally, for an integer $q \geq 2$ the *$q$-ary entropy function* is defined for $x \in [0,1]$ by

$$\begin{aligned} H_q(x) &= x\log_q(q-1) - x\log_q(x) - (1-x)\log_q(1-x) \\ &= \frac{H_2(x)}{\log_2(q)} + x\log_q(q-1) . \end{aligned}$$

We note that $H_q$ attains its maximum at $x = 1 - 1/q$. We define its inverse function, $H_q^{-1} : [0,1] \to [0, 1-1/q]$ to be such that $H_q^{-1}(H_q(x)) = x$ for any $x \in [0, 1-1/q]$. We define the following error function for $q \geq 2$ and $0 \leq r \leq k$,

$$\begin{aligned} \epsilon(q,k,r) = \log_q(e)\left(\frac{1}{12r} + \frac{1}{12(k-r)} - \frac{1}{12k+1}\right) \\ + \frac{1}{2}\log_q\left(\frac{2\pi r(k-r)}{k}\right) . \end{aligned} \tag{5}$$

The following lemmas, the proofs of which are given in Section 3.4, are key to deriving numerical bounds. The first gives both upper and lower bounds on the size of a Hamming ball.

Lemma 3.2. *Let $k, q, r$ be integers with $q \geq 2$ and $0 \leq r \leq k$. Then,*

$$q^{kH_q(r/k) - \epsilon(q,k,r)} \leq B_{q,k}(r) . \tag{6}$$

*Additionally, if $0 \leq r \leq k(1-1/q)$,*

$$B_{q,k}(r) \leq q^{kH_q(r/k)} . \tag{7}$$

The second lemma lower bounds the value of $\mathrm{rd}_{q,k}(N)$.

Lemma 3.3. *Let $N, q, k$ be positive integers such that $q \geq 2$ and $N \leq q^k$. Then,*

$$\left\lfloor H_q^{-1}\left(\frac{\log_q(N)}{k}\right) \cdot k \right\rfloor \leq \mathrm{rd}_{q,k}(N).$$

The following provides a two-sided bound on $H_q^{-1}$:

Lemma 3.4. *Let $q \geq 2$ be an integer, and $x \in [0,1]$ a real number. Then,*

$$\min(x, 1 - \frac{1}{q}) - \frac{1}{\log_2(q)} \leq H_q^{-1}(x) \leq x(1 - \frac{1}{q}) .$$

These bounds are good when $q$ is large.

*Deriving numerical bounds.* We now use the above to derive upper bounds for example parameter values. Let $b \geq 1$ be a block size, so that the alphabet has size $q = 2^b$. Fix some big-key length $k^*$ (in bits) and leakage length $\ell^*$ (in bits) that are multiples of $b$, and let $k = k^*/b$ and $\ell = \ell^*/b$ be the big-key and leakage lengths, respectively, in blocks. We assume that $\tau$ and $\ell$ satisfy that $\ell \geq \tau$, as the below method only apply when this condition is met. We note that this is a reasonable assumption for practical applications,

as leakage length $\ell$ is usually large, and we are attempting to keep the probe complexity, $\tau$, small. Now, suppose we have obtained some integer value $r$ such that: (1) $r \le \mathrm{rd}_{q,k}(q^{k-\ell})$ and (2) $0 \le r \le (k-\tau)(1-1/q)$. Then, we use Equation (6) to lower bound $B_{q,k}(r)$. Given condition (2), we can use Equation (7) to upper bound the quantity $B_{q,k-\tau}(r)$. This results in an upper bound, denoted $\mathrm{RatioBound}_{q,k,\ell,\tau}(r)$, for the ratio $B_{q,k-\tau}(r)/B_{q,k}(r)$:

$$\mathrm{RatioBound}_{q,k,\tau}(r) = \frac{q^{(k-\tau)H_q(r/(k-\tau))}}{q^{kH_q(r/k)-\epsilon(q,k,r)}} .$$

Note that in the above expression, the terms $H_q(r/(k-\tau))$, $H_q(r/k)$ and $\epsilon(q,k,r)$ can be computed numerically for any given value of $q, k, \tau$ and $r$. Hence, deriving numerical upper bound for the ratio $B_{q,k-\tau}(r)/B_{q,k}(r)$ amounts to obtaining a value $r$ satisfying the two conditions given above. We take $r$ to be $r_{q,k,\ell}$, defined as

$$r_{q,k,\ell} = \left\lfloor H_q^{-1}(\frac{k-\ell}{k}) \cdot k \right\rfloor .$$

Here, we assume that a method of obtaining numerical lower bounds for $H_q^{-1}(x)$ is available[1]. We now check the two conditions required. For condition (1), we know that $r_{q,k,\ell} \le \mathrm{rd}_{q,k}(q^{k-\ell})$ by Lemma 3.3 (taking $N = q^{k-\ell}$). For condition (2), note that by Lemma 3.4 and the assumption that $\ell \ge \tau$,

$$H_q^{-1}(\frac{k-\ell}{k}) \le \frac{k-\ell}{k}(1-1/q) \le \frac{k-\tau}{k}(1-1/q) .$$

Hence,

$$r_{q,k,\ell} = \left\lfloor H_q^{-1}(\frac{k-\ell}{k}) \cdot k \right\rfloor \le (k-\tau)(1-1/q) .$$

We consider the quantity

$$\overline{\mathbf{Adv}}^{\mathrm{skp}}_{q,k,\tau}(\ell) = \mathrm{RatioBound}_{q,k,\ell,\tau}(r_{q,k,\ell}) . \tag{8}$$

We note that since $r = r_{q,k,\ell}$ satisfies condition (1) and (2), by Theorem 3.1 and above analysis,

$$\mathbf{Adv}^{\mathrm{skp}}_{q,k,\tau}(\ell) \le \frac{B_{q,k-\tau}(r_{q,k,\ell})}{B_{q,k}(r_{q,k,\ell})} \le \overline{\mathbf{Adv}}^{\mathrm{skp}}_{q,k,\tau}(\ell) .$$

Hence, $\overline{\mathbf{Adv}}^{\mathrm{skp}}_{q,k,\tau}(\ell)$ is an upper bound for $\mathbf{Adv}^{\mathrm{skp}}_{q,k,\tau}(\ell)$. Now, given a particular desired security level, $s$, we want to find the smallest $\tau$ such that $\overline{\mathbf{Adv}}^{\mathrm{skp}}_{q,k,\tau}(\ell) \le 2^{-s}$. We let

$$\overline{\mathbf{Probes}}_{k^*,\ell^*,b}(s) = \min\left\{\tau \in [k+1] \ : \ \overline{\mathbf{Adv}}^{\mathrm{skp}}_{q,k,\tau}(\ell) \le 2^{-s}\right\} .$$

Note that this is similar to the definition of $\mathbf{Probes}_{k^*,\ell^*,b}(s)$ (Equation (3)), only that $\mathbf{Adv}^{\mathrm{skp}}_{q,k,\tau}(\ell)$ is replaced with $\overline{\mathbf{Adv}}^{\mathrm{skp}}_{q,k,\tau}(\ell)$. Thence,

$$\mathbf{Probes}_{k^*,\ell^*,b}(s) \le \overline{\mathbf{Probes}}_{k^*,\ell^*,b}(s) . \tag{9}$$

We note that $\overline{\mathbf{Probes}}_{k^*,\ell^*,b}(s)$ can be computed numerically by iteratively incrementing $\tau$ and computing $\overline{\mathbf{Adv}}^{\mathrm{skp}}_{q,k,\tau}(\ell)$. Fig. 1 shows values of $\overline{\mathbf{Probes}}_{k^*,\ell^*,b}(s)$ for various practical values of $k^*$, $\ell^*$, $b$ and $s$.

[1]For example, this is available in mathematical software Sage. Also, when $q$ is large, Lemma 3.4 provides a good lower bound for $H_q^{-1}$ that is easily computed numerically.

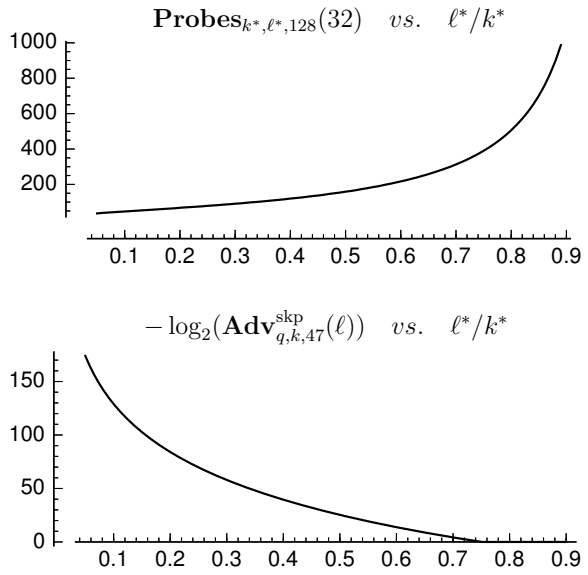


**Figure 3: Fix the big key length $k^*$ to be 100 GBytes. The top graph plots (an upper bound on) $\mathbf{Probes}_{k^*,\rho k^*,128}(32)$ as a function of the leakage rate $\rho$. The bottom graph plots (a lower bound on) $-\log_2(\mathbf{Adv}^{\mathrm{skp}}_{2^{32},k,47}(\rho k))$ as a function of $\rho$, where $k = k^*/32$.**

*Plots.* For the top plot, we fix the following:

- Blocksize $b = 32$ bits, so that $q = 2^{32}$.
- Leakage length $\ell^* = 8 \cdot 10^{10}$ bits = 10 GBytes, so that $\ell = \ell^*/32$.
- Desired security level $s = 128$ bits.

The top graph in Fig. 3 plots $\overline{\mathbf{Probes}}_{\ell^*/\rho,\ell^*,b}(s)$, upper bound for $\mathbf{Probes}_{\ell^*/\rho,\ell^*,s}(b)$, as a function of the leakage rate $\rho$. The top plot shows that the number of probes needed to maintain $s$ bits of security increases faster once the leakage rate goes over 50%. Hence, for applications, it may be beneficial to use big keys that are big enough so that the leakage rate can be assumed to be less than 50%. For example, if 10 GBytes is the leakage bound, one might, for efficiency, target big key of size at least 20 GBytes.

For the bottom plot, we fix the following

- Blocksize $b = 32$ bits, so that $q = 2^{32}$.
- Big key length $k^* = 8 \cdot 10^{11}$ bits = 100 GBytes, so that $k = k^*/32$.
- Number of probes $\tau = 47$.

The number 47 has been chosen because, as per Fig. 1, it ensures $\mathbf{Adv}^{\mathrm{skp}}_{q,k,\tau}(k/10) \le 2^{-128}$. Now with $b, k^*, \tau$ (and thus also $q, k$) fixed, the bottom graph plots $-\log_2(\overline{\mathbf{Adv}}^{\mathrm{skp}}_{q,k,\tau}\rho \cdot k)$, lower bound for $-\log_2(\mathbf{Adv}^{\mathrm{skp}}_{q,k,\tau}(\rho \cdot k))$, as a function of leakage rate $\rho$. The bottom plot in Fig. 3 demonstrates that, even though a scheme is designed for 10% leakage, security degrades gradually as the leakage rate goes over 10%.

### 3.3 Proof of Theorem 3.1

We follow the framework of the proof of BKR [5].

*Restricted subkey prediction.* The proof involves consideration of a simpler game, called the restricted subkey prediction game, denoted $\mathbf{G}^{\mathrm{rskp}}$ and shown on the right in Fig. 2. Game $\mathbf{G}^{\mathrm{rskp}}$ is similar to game $\mathbf{G}^{\mathrm{skp}}$, except that there is no leakage function Lk and leakage $L$. Instead, the big key $\boldsymbol{K}$ is drawn from a restricted subset $\mathcal{K} \subseteq [q]^k$ of big keys. We define the following advantage metrics:

$$\begin{aligned}
\mathbf{Adv}^{\mathrm{rskp}}_{k,\tau}(\mathcal{A},\mathcal{K}) &= \Pr\left[\mathbf{G}^{\mathrm{rskp}}_{k,\tau}(\mathcal{A},\mathcal{K})\right],\\
\mathbf{Adv}^{\mathrm{rskp}}_{k,\tau}(\mathcal{K}) &= \max_{\mathcal{A}} \mathbf{Adv}^{\mathrm{rskp}}_{k,\tau}(\mathcal{A},\mathcal{K}),\\
\mathbf{Adv}^{\mathrm{rskp}}_{q,k,\tau}(N) &= \max_{\mathcal{K}\subseteq[q]^k,|\mathcal{K}|=N} \mathbf{Adv}^{\mathrm{rskp}}_{k,\tau}(\mathcal{K}).
\end{aligned}$$

The first advantage is the probability that the game outputs true, meaning the probability that the adversary successfully returns $\boldsymbol{K}[\mathbf{p}]$. The second advantage is obtained by maximizing the first one over all adversaries $\mathcal{A}$. The third advantage is obtained by maximizing the second advantage over all sets $\mathcal{K} \subseteq [q]^k$ that have size $N$. We note that the first two advantages do not have $q$ in the subscript, which is due to the fact that $\mathcal{K}$ encodes the value of $q$.

*Monotone sets.* Let $x, x'$ be vectors in $[q]^k$. We say that $x$ dominates $x'$, or $x'$ is dominated by $x$, written $x' \le x$, if $x'$ can be obtained by changing non-zero coordinates of $x$ to 0. We let

$$\mathrm{DS}_{q,k}(x) = \{x' \in [q]^k \,:\, x' \le x\}$$

be the set of all $x'$ dominated by $x$. A set $\mathcal{K} \subseteq [q]^k$ is *monotone* if

$$\textstyle\bigcup_{x\in\mathcal{K}} \mathrm{DS}_{q,k}(x) \subseteq \mathcal{K}\ .$$

That is, if $x \in \mathcal{K}$, and $x'$ is dominated by $x$, then $x' \in \mathcal{K}$. For example, a Hamming ball in $[q]^k$, of any radius, is a monotone set.

*Some notation.* For integers $x, \tau \ge 0$, we let

$$x_{(\tau)} = \prod_{i=0}^{\tau-1}(x-i) = \prod_{j=x-\tau+1}^{x} j\ . \tag{10}$$

Notice that $x_{(\tau)} = 0$ if $\tau > x$. This can be seen because, if $\tau > x$, then, in the second product above, the starting value for $j$ is $\le 0$, and since $x \ge 0$, this means the term $j = 0$ is included in the product. Also when $\tau = 0$, the product has zero terms, and hence by convention takes value 1, meaning $x_{(0)} = 1$ for all $x \ge 0$. We use below the notation from Equation (10).

For a nonempty $\mathcal{K} \subseteq [q]^k$, we define the function

$$g_{k,\tau}(\mathcal{K}) = \frac{1}{|\mathcal{K}|}\sum_{x\in\mathcal{K}} \frac{(k-w(x))_{(\tau)}}{k_{(\tau)}}\ . \tag{11}$$

The following lemma says that if $\mathcal{K}$ is monotone, then the restricted subkey prediction advantage for big keys drawn from $\mathcal{K}$ can be expressed *exactly*, and in particular by the function of Equation (11).

Lemma 3.5. *Let $q, \tau, k$ be positive integers such that $\tau \le k$ and $q \ge 2$. Let $\mathcal{K} \subseteq [q]^k$ be a non-empty monotone set. Then,*

$$\mathbf{Adv}^{\mathrm{rskp}}_{k,\tau}(\mathcal{K}) = g_{k,\tau}(\mathcal{K}).$$

Proof (of Lemma 3.5). Let $\mathcal{A}_0$ be the adversary that, on input $\mathbf{p}$, always returns the all-0 $\tau$-vector. We claim that this adversary maximizes the advantage, meaning

$$\mathbf{Adv}^{\mathrm{rskp}}_{k,\tau}(\mathcal{K}) = \mathbf{Adv}^{\mathrm{rskp}}_{k,\tau}(\mathcal{K},\mathcal{A}_0)\ .$$

This follows from the assumption that $\mathcal{K}$ is monotone. Now, we compute the advantage of $\mathcal{A}_0$. For $\boldsymbol{K} \in [q]^k$, let $\mathrm{Z}(\boldsymbol{K})$ denote the set of all $\mathbf{p} \in [k]^{(\tau)}$ such that $\mathbf{K}[\mathbf{p}] = (0,\ldots,0)$. We have

$$\begin{aligned}
\mathbf{Adv}^{\mathrm{rskp}}_{k,\tau}(\mathcal{K},\mathcal{A}_0) &= \frac{1}{|\mathcal{K}|}\sum_{\boldsymbol{K}\in\mathcal{K}} \frac{|\mathrm{Z}(\boldsymbol{K})|}{|[k]^{(\tau)}|}\\
&= \frac{1}{|\mathcal{K}|}\sum_{\boldsymbol{K}\in\mathcal{K}} \frac{(k-w(\mathbf{K}))_{(\tau)}}{k_{(\tau)}}\\
&= g_{k,\tau}(\mathcal{K})\ .
\end{aligned}$$

□

We say that a set $\mathcal{K} \subseteq [q]^k$ is *sandwiched between hamming balls* if

$$B_{q,k}(r) \subseteq \mathcal{K} \subset B_{q,k}(r+1)$$

for $r = \mathrm{rd}_{q,k}(|\mathcal{K}|)$. For $N$ an integer such that $1 \le N \le q^k$, we define

$$\begin{aligned}
G_{q,k,\tau}(N) = \frac{1}{N}&\sum_{i=0}^{\mathrm{rd}_{q,k}(N)} (q-1)^i\binom{k}{i}\frac{(k-i)_{(\tau)}}{k_{(\tau)}}\\
&+\left(1-\frac{B_{q,k}(\mathrm{rd}_{q,k}(N))}{N}\right)\frac{(k-(\mathrm{rd}_{q,k}(N)+1))_{(\tau)}}{k_{(\tau)}}\ .
\end{aligned} \tag{12}$$

The following says that if $\mathcal{K}$ is monotone and sandwiched between Hamming balls, then the restricted subkey prediction advantage for big keys drawn from $\mathcal{K}$ can be expressed *exactly*, and in particular by the function of Equation (12).

Lemma 3.6. *Let $q, \tau, k$ be positive integers such that $\tau \le k$ and $q \ge 2$. Let $\mathcal{K} \subseteq [q]^k$ be a non-empty monotone set that is also sandwiched between hamming balls, i.e. $\mathbf{B}_{q,k}(r) \subseteq \mathcal{K} \subset \mathbf{B}_{q,k}(r+1)$ for $r = \mathrm{rd}_{q,k}(|\mathcal{K}|)$. Then*

$$\mathbf{Adv}^{\mathrm{rskp}}_{k,\tau}(\mathcal{K}) = G_{q,k,\tau}(|\mathcal{K}|)\ .$$

Proof (of Lemma 3.6). Let $N = |\mathcal{K}|$. By Lemma 3.5, we have

$$\mathbf{Adv}^{\mathrm{skp}}_{k,\tau}(\mathcal{K}) = \frac{1}{N}\sum_{x\in\mathcal{K}} \frac{(k-w(x))_{(\tau)}}{k_{(\tau)}}\ .$$

Since $\mathbf{B}_{q,k}(r) \subseteq \mathcal{K} \subset \mathbf{B}_{q,k}(r+1)$. This means $\mathbf{B}_{q,k}(i) \subseteq \mathcal{K}$ for $i = 0,\ldots,r$, and $\mathcal{K}$ contains $N - B_{q,k}(r)$ vectors of weight $r+1$. Thus, the above equals

$$\begin{aligned}
&\frac{N-B_{q,k}(r)}{N}\frac{(k-r-1)_{(\tau)}}{k_{(\tau)}} + \frac{1}{N}\sum_{i=0}^{r}(q-1)^i\binom{k}{i}\frac{(k-i)_{(\tau)}}{k_{(\tau)}}\\
&= G_{q,k,\tau}(N)
\end{aligned}$$

as claimed. □

Next, we show that monotone sets sandwiched between Hamming balls are the extremal cases for the restricted subkey prediction game, meaning that they maximize the restricted subkey prediction advantage. The following is analogous to [5, Lemmas 6,8]. We streamline their analysis and extend it to large alphabets.

Lemma 3.7. *Let $q, k, N$ be positive integers. Suppose $q \geq 2$, $N \leq q^k$ and $\tau \leq k$. Then, there is a non-empty monotone set $\mathcal{K} \subseteq [q]^k$ of size $N$ such that*

$$\mathbf{Adv}^{\mathrm{rskp}}_{q,k,\tau}(N) = \mathbf{Adv}^{\mathrm{rskp}}_{k,\tau}(\mathcal{K}) .$$

*Additionally, $\mathcal{K}$ is also sandwiched between hamming balls, i.e. for $r = \mathrm{rd}_{q,k}(N)$,*

$$\mathbf{B}_{q,k}(r) \subseteq \mathcal{K} \subset \mathbf{B}_{q,k}(r+1) .$$

The proof of Lemma 3.7 is deferred to Section 3.3.1. As a direct corollary of Lemma 3.6 and Lemma 3.7, we get the following result.

Corollary 3.8. *Let $q, \tau, k$ be positive integers such that $\tau \leq k$ and $q \geq 2$. Then,*

$$\mathbf{Adv}^{\mathrm{rskp}}_{q,k,\tau}(N) = G_{q,k,\tau}(N) . \tag{13}$$

Hence, from this point on, we identify the two functions $\mathbf{Adv}^{\mathrm{rskp}}_{q,k,\tau}(\cdot)$ and $G_{q,k,\tau}(\cdot)$. Next, we observe a useful property of $G_{q,k,\tau}(N)$. In particular, it is decreasing in the domain $[1..q^k]$.

Lemma 3.9. *Let $q, \tau, k$ be positive integers such that $\tau \leq k$ and $q \geq 2$. Let $i, j$ be integers such that $1 \leq i \leq j \leq q^k$. Then,*

$$G_{q,k,\tau}(i) \geq G_{q,k,\tau}(j) .$$

We proceed to relate the restricted subkey-prediction game to the subkey-prediction game via the lemma below.

Lemma 3.10. *Let $\ell, q, k, \tau$ be integers such that $0 \leq \ell \leq k$, $q \geq 2$, and $1 \leq \tau \leq k$. Then,*

$$\mathbf{Adv}^{\mathrm{skp}}_{q,k,\tau}(\ell) \leq \mathbf{Adv}^{\mathrm{rskp}}_{q,k,\tau}(q^{k-\ell}) .$$

The proofs of Lemma 3.9 and Lemma 3.10 are deferred to Section 3.3.2. Finally, we give a way to bound the expression $G_{q,k,\tau}(N)$. In particular, we show that it is at most the ratio of two hamming balls of the same radius $\mathrm{rd}_{q,k}(N)$; one with dimension $k - \tau$ and one with dimension $k$. Recall that BKR did not give concrete numerical upper bounds for their subkey-prediction advantage, only *estimates*. Due to assuming the uniqueness of probes, we are able to simplify our expression $G_{q,k,\tau}(N)$. In particular, we note that for non-negative integers $k, i, \tau$ such that $i, \tau \leq k$,

$$\binom{k}{i}\frac{(k-i)_{(\tau)}}{k_{(\tau)}} = \frac{k_{(i)}}{i_{(i)}} \cdot \frac{(k-i)_{(\tau)}}{k_{(\tau)}} = \frac{(k-\tau)_{(i)}}{i_{(i)}} = \binom{k-\tau}{i}. \tag{14}$$

This property allows us to prove the following lemma.

Lemma 3.11. *Let $N, q, k, \tau, r$ be positive integers such that $q \geq 2$, $N \leq q^k$, $\tau \leq k$ and $r \leq \mathrm{rd}_{q,k}(N)$. Then*

$$G_{q,k,\tau}(N) \leq \frac{B_{q,k-\tau}(r)}{B_{q,k}(r)}.$$

Proof (of Lemma 3.11). By Lemma 3.9,

$$G_{q,k,\tau}(N) \leq G_{q,k,\tau}(B_{q,k}(r)).$$

By Equation (12) and Equation (14),

$$\begin{aligned} G_{q,k,\tau}(B_{q,k}(r)) &\leq \frac{1}{B_{q,k}(r)} \sum_{i=0}^{\mathrm{rd}_{q,k}(B_{q,k}(r))} (q-1)^i \binom{k}{i} \frac{(k-i)_{(\tau)}}{k_{(\tau)}} \\ &= \frac{1}{B_{q,k}(r)} \sum_{i=0}^{r} (q-1)^i \binom{k-\tau}{i} \\ &= \frac{B_{q,k-\tau}(r)}{B_{q,k}(r)} . \end{aligned}$$

□

The proof of Theorem 3.1 follows directly.

Proof. (of Theorem 3.1). Note that when $r = 0$, Equation (4) is trivially true. Hence, we let $r \leq \mathrm{rd}_{q,k}(N)$ be a positive integer. Then,

$$\begin{aligned} \mathbf{Adv}^{\mathrm{skp}}_{q,k,\tau}(l) &\leq \mathbf{Adv}^{\mathrm{rskp}}_{q,k,\tau}(q^{k-l}) && \text{(Lemma 3.10)} \\ &= G_{q,k,\tau}(q^{k-l}) && \text{(Corollary 3.8)} \\ &\leq \frac{B_{q,k-\tau}(r)}{B_{q,k}(r)} . && \text{(Lemma 3.11)} \end{aligned}$$

□

### 3.3.1 Proof of Lemma 3.7.

Proof (of Lemma 3.7). Let

$$\mathcal{T} = \left\{ \mathcal{K} \subseteq [q]^k \ : \ |\mathcal{K}| = N \text{ and } \mathbf{Adv}^{\mathrm{rskp}}_{k,\tau}(\mathcal{K}) = \mathbf{Adv}^{\mathrm{rskp}}_{q,k,\tau}(N) \right\}.$$

Let $\mathcal{K} \in \mathcal{T}$ be the minimal weight element, i.e. the element $\mathcal{K} \in \mathcal{T}$ that minimizes the value $w(\mathcal{K}) = \sum_{x \in \mathcal{K}} w(x)$. We will show that $\mathcal{K}$ is a set satisfying the properties claimed in the lemma. We will prove the two properties separately, namely that $\mathcal{K}$ is monotone and $B_{q,k}(r) \subseteq \mathcal{K} \subset B_{q,k}(r+1)$. We first claim that $\mathcal{K}$ is monotone. The idea is to define a "shifting" operation for any set $\mathcal{K}' \subseteq [q]^k$ at a coordinate to increase $\mathbf{Adv}^{\mathrm{rskp}}_{k,\tau}(\mathcal{K}')$ while decreasing $w(\mathcal{K}')$. Seeking a contradiction, suppose $\mathcal{K}$ is not monotone. Without loss of generality, suppose that for all pairs of $x \in \mathcal{K}$ and $y \notin \mathcal{K}$ such that $y \leq x$, we have that $x$ and $y$ differ only in the first component. We build another set $\mathcal{K}'$ with the following properties.

(1) $|\mathcal{K}'| = |\mathcal{K}|$
(2) $w(\mathcal{K}') \leq w(\mathcal{K})$
(3) $\mathbf{Adv}^{\mathrm{rskp}}_{k,\tau}(\mathcal{K}') \geq \mathbf{Adv}^{\mathrm{rskp}}_{k,\tau}(\mathcal{K})$

We first explain briefly explain the construction of $\mathcal{K}'$ on the high level before giving the formal construction. Let $z \in [q]^{k-1}$. We will attempt to "swap" vectors of the form $\alpha \| z$, for $\alpha \in [q]$, in and out of $\mathcal{K}$. The swapping is done in two cases. We define $D_z$ to contain the $\alpha$'s such that $\alpha \| z \in \mathcal{K}$. First, if $0 \in D_z$ or $D_z = \emptyset$, no swapping will be done. Second, if $0 \notin D_z$ and $D_z \neq \emptyset$, then we will do the following. Let $\beta = \max D_z$. We will remove the element $\beta \| z$ from $\mathcal{K}$ and add the element $0 \| z$ to $\mathcal{K}$. After such operations are done for all $z \in [q]^{k-1}$, the resulting set will be $\mathcal{K}'$. Formally, the construction of $\mathcal{K}'$ is given below. $\mathcal{K}'$ is constructed from $\mathcal{K}$ via the function $\phi : [q]^k \to [q]^k$, which is defined relative to the set $B$ (set $A$ is used in the later analysis). Sets $A$ and $B$ partition the set of

strings of length $k-1$. Set $A$ consists of $z$'s such that no swapping will be done. Set $B$ consists of $z$'s such that swapping will be done. The formal definition for $A, B, \phi$, and $\mathcal{K}'$ is as follows:

$$A = \left\{ z \in [q]^{k-1} : 0 \in D_z \text{ or } D_z = \emptyset \right\},$$
$$B = \left\{ z \in [q]^{k-1} : 0 \notin D_z \text{ and } D_z \neq \emptyset \right\},$$
$$\phi(\alpha \| z) = \begin{cases} 0 \| z & \text{if } z \in B \text{ and } \alpha = \max D_z \\ (\max D_z) \| z & \text{if } z \in B \text{ and } \alpha = 0 \\ \alpha \| z & \text{otherwise} \end{cases},$$
$$\mathcal{K}' = \left\{ \phi(x) : x \in \mathcal{K} \right\}.$$

By construction, we note that the swapping operation preserves the size of the set and only decreases its overall weight. Hence, $|\mathcal{K}'| = |\mathcal{K}|$ and $w(\mathcal{K}') \leq w(\mathcal{K})$. It remains to show property (3). Let $\mathcal{A}$ be an adversary such that $\mathbf{Adv}^{\text{rskp}}_{k,\tau}(\mathcal{A}, \mathcal{K}) = \mathbf{Adv}^{\text{rskp}}_{k,\tau}(\mathcal{K})$. Consider the adversary $\mathcal{A}'$ that behaves exactly as $\mathcal{A}$ with the exception that it always guess 0 for the first position. More precisely, $\mathcal{A}'$ does the following.

Adversary $\mathcal{A}'((s_1, \ldots, s_\tau))$
$J' \leftarrow \mathcal{A}((s_1, \ldots, s_\tau))$
For $i \leftarrow 1, \ldots, \tau$ do
  If $s_i = 1$ then $J'[i] \leftarrow 0$
Return $J'$

Let $P(\cdot)$ denote the probability function in game $\mathbf{G}^{\text{rskp}}_{k,\tau}(\mathcal{A}, \mathcal{K})$ and $P'(\cdot)$ the probability function in game $\mathbf{G}^{\text{rskp}}_{k,\tau}(\mathcal{A}', \mathcal{K}')$. We now define three events for both games $\mathbf{G}^{\text{rskp}}_{k,\tau}(\mathcal{A}, \mathcal{K})$, $\mathbf{G}^{\text{rskp}}_{k,\tau}(\mathcal{A}', \mathcal{K}')$, where $z \in [q]^{k-1}$.

| | | |
|---|---|---|
| WIN | : | The game returns true |
| ONE | : | $1 \in \{s_1, \ldots, s_\tau\}$ |
| $\text{S}_z$ | : | $(\mathbf{K}[1..k] = z)$, ONE and $(\forall i, s_i \neq 1 : J'[i] = \mathbf{K}[s_i])$ |

Note that $P(\text{ONE}) = P'(\text{ONE})$, and $P(\text{WIN} | \neg\text{ONE}) = P'(\text{WIN} | \neg\text{ONE})$. We claim that $P(\text{WIN} \mid \text{ONE}) \leq P'(\text{WIN} \mid \text{ONE})$. If so we have

$$\begin{aligned} &\mathbf{Adv}^{\text{rskp}}_{k,\tau}(\mathcal{A}, \mathcal{K}) \\ &= P(\text{WIN}) \\ &= P(\text{WIN} \mid \text{ONE}) \cdot P(\text{ONE}) + P(\text{WIN} \mid \neg\text{ONE}) \cdot P(\neg\text{ONE}) \\ &= P(\text{WIN} \mid \text{ONE}) \cdot P'(\text{ONE}) + P'(\text{WIN} \mid \neg\text{ONE}) \cdot P'(\neg\text{ONE}) \\ &\leq P'(\text{WIN} \mid \text{ONE}) \cdot P'(\text{ONE}) + P'(\text{WIN} \mid \neg\text{ONE}) \cdot P'(\neg\text{ONE}) \\ &= P'(\text{WIN}) = \mathbf{Adv}^{\text{rskp}}_{k,\tau}(\mathcal{A}', \mathcal{K}') . \end{aligned}$$

So now we need to show that $P(\text{WIN} \mid \text{ONE}) \leq P'(\text{WIN} \mid \text{ONE})$. We have

$$\begin{aligned} P(\text{WIN} \mid \text{ONE}) &= \sum_{z \in [q]^{k-1}} P(\text{WIN} \mid \text{S}_z) \cdot P(\text{S}_z) \\ &= \sum_{z \in [q]^{k-1}} P(\text{WIN} \mid S_z) \cdot P'(S_z) \qquad (15) \\ &\leq \sum_{z \in [q]^{k-1}} P'(\text{WIN} \mid S_z) \cdot P'(S_z) \qquad (16) \\ &= P'(\text{WIN} \mid \text{ONE}) . \end{aligned}$$

Equation (15) is true because $P(\text{S}_z) = P'(\text{S}_z)$ for all $z \in [q]^{k-1}$, since the swapping operation do not change the last $k-1$ component of any vector. Next, we argue the validity of Equation (16). Let $z \in [q]^{k-1}$ such that $P(S_z) \neq 0$ (and hence $P'(S_z) \neq 0$), which means that there is some $\alpha \in [q]$ such that $\alpha \| z \in \mathcal{K}$. For any $z \in [q]^{k-1}$, consider the sets $U_z = \{\alpha \in [q] : \alpha \| z \in \mathcal{K}\}$ and $V_z = \{\alpha \in [q] : \alpha \| z \in \mathcal{K}'\}$. Note that $P(\text{WIN} \mid \text{S}_z) \leq 1/|U_z|$ and $P'(\text{WIN} \mid \text{S}_z) \leq 1/|V_z|$. Additionally, we note that $|U_z| = |V_z|$, and $V_z$ always contains 0. Since $\mathcal{A}'$ always guess 0 for the first component, we have $P'(\text{WIN} \mid \text{S}_z) = 1/|V_z|$. Therefore,

$$P(\text{WIN} \mid \text{S}_z) \leq \frac{1}{|U_z|} = \frac{1}{|V_z|} = P'(\text{WIN} \mid \text{S}_z) .$$

Next, we show that $\mathcal{K}$ must be sandwiched between two hamming balls. We first claim that $\mathbf{B}_{q,k}(r) \subseteq \mathcal{K}$. Seeking a contradiction, suppose that $\mathbf{B}_{q,k}(r) \nsubseteq \mathcal{K}$. Let $x'$ be a point in $\mathbf{B}_{q,k}(r) \setminus \mathcal{K}$ of minimal Hamming weight. Let $x$ be a point in $\mathcal{K} \setminus \mathbf{B}_{q,k}(r)$ of maximal Hamming weight. We claim that $w(x) > w(x')$, otherwise $\mathbf{B}_{q,k}(r) \subseteq \mathcal{K}$. Let $\mathcal{K}'$ be obtained by removing $x$ from $\mathcal{K}$ and then adding $x'$, i.e. $\mathcal{K}' = (\mathcal{K} \setminus \{x\}) \cup \{x'\}$. Because $x'$ was minimal in Hamming weight and $x$ was maximal in Hamming weight, the set $\mathcal{K}'$ continues to be monotone, and it has size $N$. Also $g_{k,\tau}(\mathcal{K}) < g_{k,\tau}(\mathcal{K}')$ because $w(x) > w(x')$. Hence, by Lemma 3.5

$$\mathbf{Adv}^{\text{rskp}}_{k,\tau}(\mathcal{K}) = g_{k,\tau}(\mathcal{K}) < g_{k,\tau}(\mathcal{K}') = \mathbf{Adv}^{\text{rskp}}_{q,k,\tau}(\mathcal{K}') .$$

This contradicts the assumption that $\mathbf{Adv}^{\text{rskp}}_{q,k,\tau}(N) = \mathbf{Adv}^{\text{rskp}}_{k,\tau}(\mathcal{K})$. Hence, it must be that $\mathbf{B}_{q,k}(r) \subseteq \mathcal{K}$. Now suppose $\mathcal{K} \nsubseteq \mathbf{B}_{q,k}(r+1)$. Let $x'$ be a point in $\mathbf{B}_{q,k}(r+1) \setminus \mathcal{K}$. Such a point exists because we know that $N < B_{q,k}(r+1)$. It must be that $w(x') = r+1$ since $\mathbf{B}_{q,k}(r) \subseteq \mathcal{K}$. Let $x$ be a point in $\mathcal{K} \setminus \mathbf{B}_{q,k}(r+1)$ of maximal Hamming weight. Note that $w(x) > r+1 = w(x')$. Let $\mathcal{K}'$ be obtained by removing $x$ from $\mathcal{K}$ and then adding $x'$, meaning $\mathcal{K}' = (\mathcal{K} \setminus \{x\}) \cup \{x'\}$. The set $\mathcal{K}'$ continues to be monotone, and it has size $N$. Also $g_{k,\tau}(\mathcal{K}) < g_{k,\tau}(\mathcal{K}')$ because $w(x) > w(x')$. Hence, by Lemma 3.5,

$$\mathbf{Adv}^{\text{rskp}}_{k,\tau}(\mathcal{K}) = g_{k,\tau}(\mathcal{K}) < g_{k,\tau}(\mathcal{K}') = \mathbf{Adv}^{\text{rskp}}_{k,\tau}(\mathcal{K}').$$

This contradicts the assumption that $\mathbf{Adv}^{\text{rskp}}_{q,k,\tau}(N) = \mathbf{Adv}^{\text{rskp}}_{k,\tau}(\mathcal{K})$. Hence, it must be that $\mathcal{K} \subset \mathbf{B}_{q,k}(r+1)$. □

*3.3.2 Proof of Lemma 3.9 and Lemma 3.10.* To prove Lemma 3.9 and Lemma 3.10, we recall the notion of *discrete concavity*. Suppose $F\colon [1..M] \to \mathbb{R}$. We say that $F$ is concave if $F(a+1) - F(a) \leq$

$F(b+1) - F(b)$ for all $a, b \in [1..M]$ satisfying $a \geq b$. Now suppose $t, m$ are integers with $1 \leq m \leq t$. Then we let

$$S(M, m, t) = \left\{ (x_1, \ldots, x_m) \in [1..M]^m \ : \ x_1 + \cdots + x_m = t \right\} .$$

Define $F^m \colon [1..M]^m \to \mathbb{R}$ by $F^m(x_1, \ldots, x_m) = F(x_1) + \cdots + F(x_m)$. We use the the following lemma proved by [5].

Lemma 3.12 ([5]). *Suppose $F \colon [1..M] \to \mathbb{R}$ is concave. Suppose $1 \leq m \leq t$ are integers such that $m$ divides $t$ and $t/m \in [1..M]$. Then*

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

Lemma 3.13. *The function, $F_{q,k,\tau} : [1..q^k] \to \mathbb{R}$, defined below, is concave.*

$$F_{q,k,\tau}(N) = \frac{N}{q^k} \cdot \mathbf{Adv}^{\mathrm{rskp}}_{q,k,\tau}(N).$$

Proof. Let $N_0, N_1$ be two integers such that $q^k \geq N_0 \geq N_1 \geq 1$. Consider, for $i = 0, 1$,

$$\Delta_i = F_{q,k,\tau}(N_i + 1) - F_{q,k,\tau}(N_i),$$

For $i = 0, 1$, we let $r_i$ be defined as follows. If $N_i = B_{q,k}(r)$ for some $r$, then we take $r_i$ to be the value such that $B_{q,k}(r_i) = N_i$. Otherwise, we let $r_i = \mathrm{rd}_{q,k}(N_i) + 1$. Note that we can now express $\Delta_i$ in terms of $r_i$ as follow (via Equation (12)),

$$\Delta_i = q^k \cdot \frac{(k - r_i)_{(\tau)}}{k_{(\tau)}} .$$

Since $N_0 \geq N_1$, we note that $r_0 \geq r_1$. Therefore, we have $\Delta_0 \leq \Delta_1$ and that $F_{q,k,\tau}$ is concave. $\square$

We first prove Lemma 3.9 using Lemma 3.13.

Proof (of Lemma 3.9). Note that,

$$G_{q,k,\tau}(N) = \frac{q^k \cdot F_{q,k,\tau}(N)}{N} .$$

We let $\Delta_i = q^k \cdot F_{q,k,\tau}(i+1) - q^k \cdot F_{q,k,\tau}(i)$ for all $i = 0, \ldots, q^k - 1$. We define $\Delta_0 = q^k \cdot F_{q,k,\tau}(1) = q^k \cdot G_{q,k,\tau}(1)$. Hence, by construction $G_{q,k,\tau}(i) = (\sum_{j=0}^{i-1} \Delta_i)/i$. Note that since $F_{q,k,\tau}(\cdot)$ is concave in the domain $[1..q^k]$, the sequence $\Delta_1, \ldots, \Delta_{q^k - 1}$ is non-increasing, meaning that $\Delta_i \geq \Delta_j$ whenever $1 \leq i \leq j \leq q^k - 1$. Additionally, we check that $\Delta_0 = q^k$ and $\Delta_1 \leq q^k$, hence $\Delta_0 \geq \Delta_1$. Therefore, the partial averages of the sequence $\Delta_0, \ldots, \Delta_{q^k - 1}$,

$$\Big(\sum_{j=0}^{i-1} \Delta_j\Big) \ / \ i = G_{q,k,\tau}(i) \ ,$$

is non-increasing as claimed. $\square$

Lastly, we prove Lemma 3.10 using Lemma 3.12 and 3.13.

Proof (of Lemma 3.10). Let $M = q^k$, $m = q^\ell$ and $t = q^k$. We note that the leakage function $\mathsf{Lk} : [q]^k \to [q]^\ell$ defines a partition of $[q]^k$ into $q^\ell$ sets, with each set being $\mathsf{Lk}^{-1}(L)$ for some $L \in [q]^\ell$. Hence, we can expand $\Pr[\mathbf{G}^{\mathrm{skp}}_{q,k,\tau}(\mathcal{A}, \mathsf{Lk})]$ by conditioning on the value of $L$. Suppose $[q]^\ell = \{L_1, \ldots, L_m\}$. We let $N_i = |\mathsf{Lk}^{-1}(L_i)|$. We derive

$$\begin{aligned}
&\mathbf{Adv}^{\mathrm{skp}}_{q,k,\tau}(\ell) \\
&= \max_{\mathsf{Lk}} \left( \sum_L \frac{|\mathsf{Lk}^{-1}(L)|}{q^k} \cdot \max_{\mathcal{A}} \Pr[\, \mathbf{G}^{\mathrm{skp}}_{q,k,\tau}(\mathcal{A}, \mathsf{Lk}) \ \mid \ \mathsf{Lk}(\mathbf{K}) = L \,] \right) \\
&= \max_{\mathsf{Lk}} \left( \sum_L \frac{|\mathsf{Lk}^{-1}(L)|}{q^k} \cdot \mathbf{Adv}^{\mathrm{rskp}}_{k,\tau}(\mathsf{Lk}^{-1}(L)) \right) \\
&\leq \max_{(N_1, \ldots, N_m) \in S(M,m,t)} \sum_{i=1}^{m} F_{q,k,\tau}(N_i) \\
&= \max_{(N_1, \ldots, N_m) \in S(M,m,t)} F^m_{q,k,\tau}(N_1, \ldots, N_m) \\
&= m \cdot F_{q,k,\tau}(2^{k-\ell}) \qquad (17) \\
&= m \cdot \frac{q^{k-\ell}}{q^k} \cdot \mathbf{Adv}^{\mathrm{rskp}}_{q,k,\tau}(q^{k-\ell}) = \mathbf{Adv}^{\mathrm{rskp}}_{q,k,\tau}(q^{k-\ell}) \ . \qquad (18)
\end{aligned}$$

Equation (17) is justified since $F_{q,k,\tau}$ is concave and $t/m = 2^{k-\ell}$. Equation (18) is by definition of $F$ and because $m = q^\ell$. $\square$

## 3.4 Proofs of Lemmas 3.2, 3.3, and 3.4

We need the following version of Stirling's approximation of $n!$.

Lemma 3.14. *[18] For any $n \in \mathbb{Z}^+$,*

$$\sqrt{2\pi n} \left(\frac{n}{e}\right)^n e^{\frac{1}{12n+1}} \leq n! \leq \sqrt{2\pi n} \left(\frac{n}{e}\right)^n e^{\frac{1}{12n}} .$$

We first prove Lemma 3.2.

Proof (of Lemma 3.2). We first show the lower bound Equation (6) . Notice that by definition of $H_q(r/k)$,

$$q^{kH_q(r/k)} = (q-1)^r (r/k)^{-r} (1 - r/k)^{r-k} .$$

Hence, by Lemma 3.14,

$$\begin{aligned}
B_{q,k}(r) &= \sum_{i=0}^{r} (q-1)^i \binom{k}{i} \\
&\geq (q-1)^r \frac{k!}{r!(k-r)!} \\
&\geq (q-1)^r \frac{\sqrt{2\pi k}(\frac{k}{e})^k e^{\frac{1}{12k+1}}}{\sqrt{2\pi r}(\frac{r}{e})^r e^{\frac{1}{12r}} \sqrt{2\pi(k-r)}(\frac{k-r}{e})^{k-r} e^{\frac{1}{12(k-r)}}} \\
&= q^{kH_q(r/k)} \frac{\sqrt{k} e^{\frac{1}{12k+1}}}{\sqrt{2\pi r(k-r)} e^{\frac{1}{12r}} e^{\frac{1}{12(k-r)}}} \\
&= q^{kH_q(r/k) - \epsilon(k,r)} .
\end{aligned}$$

Now, we assume that $r \le k - k/q$ and derive the upper bound, Equation (7).

$$\begin{aligned}
\frac{B_{q,k}(r)}{q^{kH_q(r/k)}} &= \frac{\sum_{i=0}^{r}(q-1)^i\binom{k}{i}}{(q-1)^r(r/k)^{-r}(1-r/k)^{r-k}} \\
&= \sum_{i=0}^{r}\binom{k}{i}(q-1)^{i-r}(r/k)^r(1-r/k)^{k-r} \\
&= \sum_{i=0}^{r}\binom{k}{i}(q-1)^i(1-r/k)^k\Big(\frac{r/k}{(q-1)(1-r/k)}\Big)^r \\
&\le \sum_{i=0}^{r}\binom{k}{i}(q-1)^i(1-r/k)^k\Big(\frac{r/k}{(q-1)(1-r/k)}\Big)^i \\
&= \sum_{i=0}^{r}\binom{k}{i}(r/k)^i(1-r/k)^{k-i} \\
&\le \sum_{i=0}^{k}\binom{k}{i}(r/k)^i(1-r/k)^{k-i} \\
&= 1\ ,
\end{aligned}$$

where the first inequality is by the fact that $r/k \le (q-1)(1-r/k)$ if $r \le k - k/q$. □

Lemma 3.3 follows from Lemma 3.2.

Proof (of Lemma 3.3). Per definition of $\mathrm{rd}_{q,k}(N)$, it suffices to show that

$$B_{q,k}(r) \le N\ ,$$

for $r = \lfloor H_q^{-1}(\log_q(N)/k)\cdot k\rfloor$. Per definition of $H_q^{-1}$, $r \le (1-1/q)\cdot k$. Hence, we can apply Equation (7) and obtain

$$B_{q,k}(r) \le q^{kH_q(r/k)} \le q^{kH_q(H_q^{-1}(\log_q(N)/k))} = N\ .$$

□

Lastly, we prove Lemma 3.4.

Proof (of Lemma 3.4). We first show the lower bound that

$$\min(x, 1-\frac{1}{q}) - \frac{1}{\log_2(q)} \le H_q^{-1}(x)\ . \tag{19}$$

Note that this is trivially true if the left-hand side of Equation (19) is negative. Hence, we suppose that the left-hand side of Equation (19) is non-negative. As noted before, $H_q$ is increasing in the domain $[0, 1-1/q]$. Additionally, note that $\min(x, 1-1/q) - 1/\log_2(q) \le 1-1/q$. Hence, it suffices to show

$$H_q\Big(\min(x, 1-\frac{1}{q}) - \frac{1}{\log_2(q)}\Big) \le x\ . \tag{20}$$

We consider two cases. Case 1, $x \le (1-1/q)$. Case 2, $(1-1/q) \le x \le 1$. We claim that both cases follow from the equation below, which holds for $x \in [\log_2(q), 1]$.

$$H_q(x - \frac{1}{\log_2(q)}) \le x\ . \tag{21}$$

Case 1 is directly implied by Equation (21). For case 2, note that the left-hand side of Equation (19) always evaluate to $1-1/q-1/\log_2(q)$. Hence, by Equation (21), $H_q(1-1/q-1/\log_2(q)) \le 1-1/q \le x$. Finally, we justify Equation (21). Recall that $H_q(x) = H_2(x)/\log_2(q) + x\log_q(q-1)$. We compute

$$\begin{aligned}
H_q(x - \frac{1}{\log_2(q)}) &= \frac{H_2(x - \frac{1}{\log_2(q)})}{\log_2(q)} + (x - \frac{1}{\log_2(q)})\log_q(q-1) \\
&\le \frac{1}{\log_2(q)} + x\log_q(q\cdot\frac{q-1}{q}) - \frac{\log_q(q-1)}{\log_2(q)} \\
&= \frac{1}{\log_2(q)} + x - x\log_q(\frac{q}{q-1}) - \frac{\log_q(q-1)}{\log_2(q)} \\
&= x + \frac{1}{\log_2(q)}(1 - \log_q(\frac{q^{(x\log_2(q))}}{(q-1)^{(x\log_2(q)-1)}})) \\
&\le x + \frac{1}{\log_2(q)}(1-\log_q(q)) \\
&= x.
\end{aligned}$$

Next, we show the upper bound that

$$H_q^{-1}(x) \le x(1-\frac{1}{q})\ . \tag{22}$$

Similar to the lower bound we just obtained, we note that it suffices to show $H_q(x(1-\frac{1}{q})) \ge x$. Let us define, for $x \in [0,1]$:

$$f(x) = \frac{x}{q}\log_q(\frac{x}{q}) - x\log_q(x) - (1-x+\frac{x}{q})\log_q(1-x+\frac{x}{q})\ .$$

We will show that $H_q(1(1-1/q)) = x + f(x)$. The derivation is as follows.

$$\begin{aligned}
&H_q(x(1-1/q)) \\
&= x(1-1/q)\log_q(q-1) - x(1-1/q)\log_q(x(1-1/q)) \\
&\quad - (1-x+x/q)\log_q(1-x+x/q) \\
&\ge x\log_q(q-1) - x/q\log_q(q-1) \\
&\quad - x(1-1/q)(\log_q(x) + \log_q(1-1/q)) \\
&\quad - (1-x+x/q)\log_q(1-x+x/q) \\
&= x\log_q(q-1) - x/q\log_q(q-1) \\
&\quad - x\log_q(x) - x\log_q(1-1/q) + x/q\log_q(x) + x/q\log_q(1-1/q) \\
&\quad - (1-x+x/q)\log_q(1-x+x/q) \\
&= x\Big(\log_q(q-1) + \log_q(q/(q-1))\Big) - x\log_q(x) \\
&\quad - x/q\Big(\log_q(q-1) + \log_q(1/x) + \log_q(q/(q-1))\Big) \\
&\quad - (1-x+x/q)\log_q(1-x+x/q) \\
&= x + x/q\log_q(x/q) - x\log_q(x) - (1-x+x/q)\log_q(1-x+x/q) \\
&= x + f(x)\ .
\end{aligned}$$

Lastly, we show that $f(x) \ge 0$ for any $x \in [0,1]$. First, we check that $f(0) = f(1) = 0$. Next, check that the second derivative of $f$,

$$f''(x) = \frac{q-1}{x(qx-q-x)} \le 0\ ,$$

is at most 0 for any $x \in [0,1]$. We omit the details of the derivative computation here. Hence, $f$ is concave over the domain $[0,1]$, with $f(0) = f(1) = 0$. Thence, $f(x) \ge 0$ for all $x \in [0,1]$. □

| Game $\mathbf{G}^{\text{key}}_{\text{KEY}}(\mathcal{A})$ | ROR( ) |
|---|---|
| $b \leftarrow\$ \{0,1\}$; $\mathbf{K} \leftarrow\$ [q]^k$ | $R \leftarrow\$ \{0,1\}^r$ |
| $(\mathsf{Lk}, \sigma) \leftarrow\$ \mathcal{A}^{\mathsf{H}}()$ | If $(b = 0)$ then $K \leftarrow\$ \{0,1\}^\kappa$ |
| $L \leftarrow \mathsf{Lk}^{\mathsf{H}}(\mathbf{K})$ | Else $K \leftarrow \mathsf{KEY}^{\mathsf{H}}(\mathbf{K}, R)$ |
| $b' \leftarrow\$ \mathcal{A}^{\text{ROR},\mathsf{H}}(L, \sigma)$ | Return $(R, K)$ |
| Return $(b' = b)$ | H($x$, Rng) |
| | If not $T[x, \text{Rng}]$ then |
| | $T[x, \text{Rng}] \leftarrow\$ \text{Rng}$ |
| | Return $T[x, n]$ |

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

Algorithm $\mathsf{XKEY}^{\mathsf{H}}_{q,k,\kappa,\tau,r}(\mathbf{K}, R)$ // $\mathbf{K} \in [q]^k$, $|R| = r$

$\mathbf{p} \leftarrow \mathsf{H}(R, \ [k]^{(\tau)})$; $J \leftarrow \mathbf{K}[\mathbf{p}]$; $K \leftarrow \mathsf{H}(R \| J, \ [k])$; Return $K$

**Figure 5: Encapsulation algorithm** XKEY**. Given a length-$k$ big-key $\mathbf{K}$ and a length-$r$ selector $R$, the algorithm returns a length-$\kappa$ subkey $K$. The value $\tau$ specifies the number of unique probes used.**

## 4 BIG-KEY SYMMETRIC ENCRYPTION

In [5], Big-Key symmetric encryption schemes are constructed modularly from Big-Key encapsulation schemes. In this section, we present a block-based big key encapsulation scheme that is more efficient than achieved previously.

*Key Encapsulation Schemes.* A (symmetric, Big-Key) encapsulation schemes, on input a big key $\mathbf{K}$ and a random string $R$, returns a (short) key $K$. The string $R$ encapsulates the short key $K$ in the sense that any party holding the big key $\mathbf{K}$ can derive $K$ from $R$. The security of a key encapsulation schemes is captured by $\mathbf{G}^{\text{key}}_{\text{KEY}}(\mathcal{A})$ (Fig. 4). In this game, a big key $\mathbf{K}$ is randomly sampled. The goal of the two-stage adversary $\mathcal{A}$ is to guess whether the real-or-random oracle, ROR, is returning real keys, derived using key encapsulation scheme KEY from randomly sampled $R$, or randomly sampled keys that is independent of $R$. In its first stage, $\mathcal{A}$ gets access to H and chooses a leakage function Lk and state $\sigma$. Next, the game computes $L \leftarrow \mathsf{Lk}^{\mathsf{H}}(\mathbf{K})$ and run the second stage of $\mathcal{A}$ with inputs $L, \sigma$ and oracles ROR and H. $\mathcal{A}$ wins the game if it successfully guesses the bit $b$. We define the following advantage of $\mathcal{A}$ against key encapsulation scheme KEY

$$\mathbf{Adv}^{\text{key}}_{\text{KEY}}(\mathcal{A}) = 2 \cdot \Pr\left[\mathbf{G}^{\text{key}}_{\text{KEY}}(\mathcal{A})\right] - 1.$$

*Our construction.* Our random oracle model construction is given in Fig. 5.

Theorem 4.1. *Let $k, b, \kappa, \tau, r \geq 1$ be integers. Let $q = 2^b$. Let* $\mathsf{KEY} = \mathsf{XKEY}_{q,k,\kappa,\tau,r}$ *be the big-key encapsulation scheme associated to them as per Fig. 5. Let $\mathcal{A}$ be an adversary making at most $t$ queries to its* ROR *oracle and leaking $\ell \cdot b$ bits. Assume the number of* H *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* H *queries made by $\mathcal{A}$ in its second stage is at most $q_2$. Then*

$$\mathbf{Adv}^{\text{key}}_{\text{KEY}}(\mathcal{A}) \ \leq \ q_2 \cdot t \cdot \mathbf{Adv}^{\text{skp}}_{q,k,\tau}(\ell) + \frac{t \cdot (2q_1 + t - 1)}{2^{r+1}} \qquad (23)$$

The proof of Theorem 4.1 is included in the full version [4].

| Algorithm $\mathbf{SE}.\mathsf{Enc}^{\mathsf{H}}(\mathbf{K}, M)$ | Algorithm $\mathbf{SE}.\mathsf{Dec}^{\mathsf{H}}(\mathbf{K}, M)$ |
|---|---|
| $R \leftarrow\$ \{0,1\}^r$; $\mathbf{K} \leftarrow \mathsf{KEY}^{\mathsf{H}}(\mathbf{K}, R)$ | $(R, C) \leftarrow \overline{C}$ |
| $C \leftarrow \mathsf{SE}.\mathsf{Enc}(\mathbf{K}, M)$; $\overline{C} \leftarrow (R, C)$ | $K \leftarrow \mathsf{KEY}^{\mathsf{H}}(\mathbf{K}, R)$ |
| Return $\overline{C}$ | $M \leftarrow \mathsf{SE}.\mathsf{Dec}(K, C)$ |
| | Return $M$ |

**Figure 6: Big-Key Symmetric Encryption Scheme [5, Section 5], SE, using a standard symmetric key encryption scheme** SE **and a key encapsulation mechanism** KEY**.**

*Sampling unique probes.* In XKEY, we have outsourced the sampling of the unique probes to the variable-range random oracle. We note that sampling from $[k]^{(\tau)}$ can be done via rejection sampling efficiently.

*Symmetric Encryption Schemes.* To obtain a (big-key) symmetric encryption scheme, one can plug our XKEY construction directly into the (big-key) symmetric encryption scheme (in Fig. 6) by BKR. For security, we omit the details here and appeal to [5, Theorem 13].

*Efficiency.* Let $k^* = 8 \cdot 10^{11} = 100$ GBytes, and $\ell^* = 10$ GBytes. Using $b = 8 \cdot 512 = 512$ Bytes, our XKEY makes roughly the same number of H queries compared to [5] but makes significantly less access into the big key $\mathbf{K}$ (43 vs. 271, Fig. 1). In practical instantiations where $\mathbf{K}$ is stored on slow storage medium (e.g. hard disk), this translate to 6x improvement in efficiency.

## 5 BIG-KEY IDENTIFICATION

*Identification schemes.* An identification scheme ID specifies the following:

- Via $\mathsf{prm} \leftarrow\$ \mathsf{ID.Pg}$, parameter generation algorithm ID.Pg generates parameter prm, which is a common input to all other algorithms.
- Via $(\mathsf{sk}, \mathsf{vk}, \mathsf{hlp}) \leftarrow\$ \mathsf{ID.Kg}(\mathsf{prm})$, key generation algorithm ID.Kg is run by the prover to generate secret key sk, corresponding verification key vk and a string hlp called the help string. The last is information that, conceptually, can be viewed as part of the public verification key vk, meaning public and available to the adversary, but to keep the verification key small, hlp is stored by the prover along with sk.
- Via $(\mathsf{com}, \mathsf{st}) \leftarrow\$ \mathsf{ID.Com}(\mathsf{prm})$, commitment algorithm ID.Com is run by the prover to generate its first message com, called the commitment, along with state information st that it saves.
- Via $\mathsf{chl} \leftarrow\$ \{0,1\}^{\mathsf{ID.Chl}}$, the verifier generates a random challenge chl to return to the prover.
- Via $\mathsf{rsp} \leftarrow \mathsf{ID.Rsp}(\mathsf{prm}, \mathsf{hlp}, \mathsf{sk}, \mathsf{st}, \mathsf{chl})$, deterministic response algorithm ID.Rsp is run by the prover to generate its response rsp.

Game $\mathbf{G}^{\mathrm{imp}}_{\mathsf{ID},\ell}(\mathcal{A})$

prm $\leftarrow\!\$$ ID.Pg; $s \leftarrow 0$
(sk, vk, hlp) $\leftarrow\!\$$ ID.Kg(prm)
st $\leftarrow\!\$$ $\mathcal{A}$.Setup$^{\mathsf{Leak}_\ell,\mathsf{Prover},\mathsf{H}}$(prm, vk, hlp)
(com, st′) $\leftarrow\!\$$ $\mathcal{A}$.Com$^{\mathsf{H}}$(st); chl $\leftarrow\!\$$ $\{0,1\}^{\mathsf{ID.Chl}}$
rsp $\leftarrow\!\$$ $\mathcal{A}$.Rsp$^{\mathsf{H}}$(prm, hlp, sk, st′, chl)
$d \leftarrow\!\$$ ID.Vrf$^{\mathsf{H}}$(prm, vk, com, chl, rsp)
Return $d$

Prover($i$, args)

If pst[$i$] = $\perp$ then // Commit
  (pcom[$i$], pst[$i$]) $\leftarrow\!\$$ ID.Com(prm) ; return pcom[$i$]
If prsp[$i$] = $\perp$ then // Response
  prsp[$i$] $\leftarrow\!\$$ ID.Rsp$^{\mathsf{H}}$(prm, hlp, sk, pst[$i$], args)
  Return prsp[$i$]
Return $\perp$

$\mathsf{Leak}_\ell(f)$

$L \leftarrow\!\$$ $f$(sk) ; $s \leftarrow s + |L|$
If $s \le \ell$ then return $L$ else return $\perp$

H($x$, Rng)

If $T[x, \mathsf{Rng}] = \perp$ then $T[x, \mathsf{Rng}] \leftarrow\!\$$ Rng
Return $T[x, \mathsf{Rng}]$

**Figure 7: Game defining security of identification scheme ID under pre-impersonation leakage.**

– Via $d \leftarrow$ ID.Vrf(prm, vk, com, chl, rsp), deterministic decision algorithm ID.Vrf returns a boolean decision $d$ for the verifier to accept or reject.

In the ROM, algorithms may have oracle access to the random oracle H. This syntax is non-asymptotic, in that there is no explicit security parameter. Correctness requires that

$$\Pr[\mathsf{Execute}_{\mathsf{ID}}(\mathsf{prm}, \mathsf{vk}, \mathsf{sk}, \mathsf{hlp})] = 1$$

for all prm $\in$ [ID.Pg] and (sk, vk, hlp) $\in$ [ID.Kg(prm)], where

Game $\mathsf{Execute}_{\mathsf{ID}}(\mathsf{prm}, \mathsf{vk}, \mathsf{sk}, \mathsf{hlp})$

(com, st) $\leftarrow\!\$$ ID.Com(prm)
chl $\leftarrow\!\$$ $\{0,1\}^{\mathsf{ID.Chl}}$
rsp $\leftarrow$ ID.Rsp(prm, hlp, sk, st, chl)
$d \leftarrow$ ID.Vrf(prm, vk, com, chl, rsp)
Return $d$

*Security of identification schemes.* We give definitions allowing concrete-security assessments. The core definition is that of adversary advantage. The notion captured is security against impersonation under active attack [6, 15] in the further presence of leakage on the secret key [2].

Let ID be an identification scheme. Let $\ell$ be an integer representing a bound (in bits) on the leakage. Let $\mathcal{A}$ be an *impersonation adversary*, made up of component algorithms $\mathcal{A}$.Setup, $\mathcal{A}$.Com, and $\mathcal{A}$.Rsp. We associate to these the game of Fig. 7. First, the parameters and keys are generated. Next, $\mathcal{A}$.Setup is run with access to a leakage oracle $\mathsf{Leak}_\ell$ a prover oracle Prover and the random oracle H. The leakage oracle takes input a function Lk from the adversary and returns leakage $L$ = Lk(sk). This oracle can be called

| Game $\mathbf{G}^{\mathrm{cdh}}_{\mathcal{G}}(\mathcal{A})$ | Game $\mathbf{G}^{\mathrm{dl}}_{\mathcal{G}}(\mathcal{A})$ |
|---|---|
| $(G, G_T, g, \mathbf{e}, p) \leftarrow \mathcal{G}$ | $(G, G_T, g, \mathbf{e}, p) \leftarrow \mathcal{G}$ |
| $x, y \leftarrow\!\$ [p]$ | $x \leftarrow\!\$ [p]$ |
| $h \leftarrow\!\$ \mathcal{A}(\mathcal{G}, g^x, g^y)$ | $x' \leftarrow\!\$ \mathcal{A}(\mathcal{G}, g^x)$ |
| Return $(h = g^{xy})$ | Return $(x = x')$ |

**Figure 8: Games $\mathbf{G}^{\mathrm{cdh}}_{\mathcal{G}}$ and $\mathbf{G}^{\mathrm{dl}}_{\mathcal{G}}$ defining the security of CDH and DL problems in $\mathcal{G}$.**

adaptively and any number of times, its code ensuring that the total number of bits returned to the adversary does not exceed $\ell$. The prover oracle allows an active attack in which the adversary, playing the role of a dishonest verifier, can generate prover instances and interact with them. The commitment and state of instance $i$ are produced by the game and stored as pcom[$i$] and pst[$i$], respectively. If instance $i$ has been activated, meaning pst[$i$] $\neq \perp$, then the adversary can submit, via args, a challenge of its choice, and obtain response prsp[$i$]. After exiting this setup phase, the adversary turns into a dishonest prover, aiming to convince the honest verifier to accept. It produces its commitment via $\mathcal{A}$.Com, receives a random challenge chl, and produces its response via $\mathcal{A}$.Rsp. The game returns the boolean decision $d$ of the verifier's decision function. We define the leakage impersonation advantage of $\mathcal{A}$ against ID to be

$$\mathbf{Adv}^{\mathrm{imp}}_{\mathsf{ID},\ell}(\mathcal{A}) = \Pr\left[\mathbf{G}^{\mathrm{imp}}_{\mathsf{ID},\ell}(\mathcal{A})\right].$$

*Groups.* We fix a *bilinear group description* $\mathcal{G} = (G, G_T, g, \mathbf{e}, p)$, where

– $p \ge 3$ is a prime number that will be the order of the groups
– $G, G_T$ are (cyclic) groups of order $p$
– $g \in G$ is a generator of $G$
– $\mathbf{e} : G \times G \to G_T$ is an efficiently computable, non-degenerate bilinear map. This means that (1) $e(g^a, g^b) = \mathbf{e}(g, g)^{ab}$ for all $a, b \in [p]$, and (2) $\mathbf{e}(g, g)$ is not the identity element of $G_T$.

We will base security on the assumed hardness of the CDH (Computational Diffie-Hellman) and DL (Discrete Logarithm) problems in $G$. The definitions are based on games $\mathbf{G}^{\mathrm{cdh}}$ and $\mathbf{G}^{\mathrm{dl}}$ in Fig. 8, associated to $\mathcal{G}$ and an adversary $\mathcal{A}$. We define the following CDH and DL advantages:

$$\mathbf{Adv}^{\mathrm{cdh}}_{\mathcal{G}}(\mathcal{A}) = \Pr[\mathbf{G}^{\mathrm{cdh}}_{\mathcal{G}}(\mathcal{A})]$$
$$\mathbf{Adv}^{\mathrm{dl}}_{\mathcal{G}}(\mathcal{A}) = \Pr[\mathbf{G}^{\mathrm{dl}}_{\mathcal{G}}(\mathcal{A})] .$$

Hardness of CDH of course implies hardness of DL. Quantitatively, given $\mathcal{A}$, one can construct $\mathcal{A}'$ with similar running time such that

$$\mathbf{Adv}^{\mathrm{dl}}_{\mathcal{G}}(\mathcal{A}) \le \mathbf{Adv}^{\mathrm{cdh}}_{\mathcal{G}}(\mathcal{A}').$$

ADW *identification scheme.* We present a variant of ADW's identification scheme [2], which uses a random oracle to derive the challenges (as considered in [2] without analysis). The scheme ID = ADW[$\mathcal{G}, k, m, \tau, r$] is parameterized by a bilinear group description $\mathcal{G}$ and positive integers $k, m, \tau, r$. We require that $m \ge 2$ and $k \ge \tau \ge 1$. Here $k$ is the number of blocks of the secret key, where each block is an $m$-dimensional vector over $\mathbb{Z}_p$, and $\tau$ is

Game $\mathbf{G}^{\text{pskp}}_{p,m,k,\tau}(\mathcal{A}, \mathsf{Lk})$
For $i \in [k]$ do $\mathsf{sk}[i] \leftarrow\!\$\ \overline{\mathbb{Z}_p^m}$
$\mathbf{p} \leftarrow\!\$\ [k]^{(\tau)}$; $e \leftarrow\!\$\ \mathbb{Z}_p$
For $j \in [m]$ do
  $\mathsf{sk}^*[j] = \sum_{i=0}^{\tau-1} (\mathsf{sk}[\mathbf{p}[i]][j]) e^i$
$L \leftarrow \mathsf{Lk}(\mathsf{sk})$; $\overline{\mathsf{sk}} \leftarrow\!\$\ \mathcal{A}(\mathbf{p}, e, L)$
Return $(\mathsf{sk}^* = \overline{\mathsf{sk}})$.

**Figure 9: Game $\mathbf{G}^{\text{pskp}}_{p,m,k,\tau}(\mathcal{A}, \mathsf{Lk})$. Where $\mathsf{Lk} : [q]^k \to [q]^\ell$ is a leakage function. $[k]^{(\tau)}$ contains the set of $\tau$-dimensional vectors over $[k]$ with distinct entries.**

the number of probes that algorithms make into the secret key. The parameter $r$ determines the challenge length, meaning we set $\mathsf{ID.chl} = r$. The algorithms ID.Pg, ID.Kg, ID.Com, ID.Rsp, ID.Vrf are given in Fig. 10.

Intuitively, the scheme consists of $k$ generalized Okamoto identification scheme [2, 17], and one instance of BLS signature scheme [9]. Each block of the secret key (in $\mathbb{Z}_p^m$) is a secret key for a generalized Okamoto identification scheme of dimension $m$. The public keys, $\mathsf{pk}[0], \ldots, \mathsf{pk}[k-1]$, of the $k$ Okamoto's identification schemes, are signed using the BLS signature scheme under signing key $s$, yielding signatures $\sigma[0], \ldots, \sigma[k-1]$. The public verification key of the identification scheme, consists only of the verification key, vk, of the BLS signature scheme. During identification, a random $\tau$ instances out of $k$ instances is chosen (via H by the verifier) and compressed via polynomial evaluation to $\mathsf{sk}^*, \mathsf{pk}^*$, and $\sigma^*$ by the prover. During response phase, the prover, in addition to answering the challenge from the Okamoto identification scheme, needs to transmit $\mathsf{pk}^*$ and $\sigma^*$ to the verifier. We note that the signing key, $s$, of the underlying signature scheme *must not* be visible to the attacker. This signing key is simply be discarded after Kg. (However, we note that, as ADW has pointed out, there are advanced uses of this key such as updating the big secret key.) The correctness of $\mathsf{ID} = \mathsf{ADW}[\mathcal{G}, k, m, \tau, r]$ is checked as follows. Let $\mathsf{prm} \in [\mathsf{ID.Pg}]$ and $(\mathsf{sk}, \mathsf{vk}, \mathsf{hlp}) \in [\mathsf{ID.Kg}(\mathsf{prm})]$. We claim that, during a honest execution of the protocol ($\mathsf{Execute}_{\mathsf{ID}}(\mathsf{prm}, \mathsf{sk}, \mathsf{vk}, \mathsf{hlp})$), the flags $A, B$ in ID.Vrf will both be set to true. $A$ is set to true because

$$\begin{aligned}\prod_{i=0}^{m-1} g_i^{z[i]} &= \prod_{i=0}^{m-1} g^{y[i]+c^*\cdot \mathsf{sk}^*[i]} \\ &= \prod_{i=0}^{m-1} g^{y[i]} \cdot (\prod_{i=0}^{m-1} g^{\mathsf{sk}^*[i]})^c \\ &= a \cdot \mathsf{pk}^{*c^*} \ .\end{aligned}$$

$B$ is set to true because

$$\begin{aligned}\mathbf{e}(\mathsf{pk}^* \prod_{i=0}^{\tau-1} \mathsf{H}(\mathbf{p}[i], G)^{e^i}, \mathsf{vk}) &= \mathbf{e}(\prod_{i=0}^{\tau-1} \mathsf{pk}[\mathbf{p}[i]]^{e^i} \prod_{i=0}^{\tau-1} \mathsf{H}(\mathbf{p}[i], G)^{e^i}, g^s) \\ &= \mathbf{e}(\prod_{i=0}^{\tau-1} ((\mathsf{pk}[\mathbf{p}[i]]\mathsf{H}(\mathbf{p}[i], G))^s)^{e^i}, g) \\ &= \mathbf{e}(\prod_{i=0}^{\tau-1} (\sigma[\mathbf{p}[i]])^{e^i}, g) \\ &= \mathbf{e}(\sigma^*, g) \ .\end{aligned}$$

Hence, $\Pr[\mathsf{Execute}_{\mathsf{ID}}(\mathsf{prm}, \mathsf{sk}, \mathsf{vk}, \mathsf{hlp})] = 1$, and ID satisfies correctness.

*Efficiency.* As pointed out in [2], the identification scheme has nice efficiency properties. First, the public key (verification key) is very short (one group element). Second, the communication costs of all phases are very small. The bulk of communication happens in the response phase, which outputs 2 group elements and $m$ elements from $\mathbb{Z}_p$. Third, the scheme has probe complexity depending $\tau$, which can be made small while preserving security. In particular, during each run of the protocol, only $\tau$ locations of the secret-key will be accessed (each location consist of $m$ elements of $\mathbb{Z}_p$). Fig. 11 demonstrates the computation and communication costs of different operations. Note that very small values of $\tau$ makes the scheme insecure. The crux of the security analysis amounts to giving a lower bound of $\tau$ for a desired security level. Here is where we make significant *concrete security* improvements over ADW.

*Concrete-security analysis.* Before we present the theorem stating the concrete security of the ADW identification scheme, we first need to define the following special subkey prediction game. The game $\mathbf{G}^{\text{pskp}}_{p,m,k,\tau}(\mathsf{Lk}, \mathcal{A})$ (Fig. 9) captures a particular type of subkey prediction game in which the subkey is interpreted as a tuple of polynomials. In this game, the adversary $\mathcal{A}$ needs to predict the value of these polynomials at a random point $e$, which is given to $\mathcal{A}$. We define the following prediction advantage

$$\mathbf{Adv}^{\text{pskp}}_{p,m,k,\tau}(\ell) = \max_{\mathcal{A}, \mathsf{Lk}:(\mathbb{Z}_p^m)^k \to (\mathbb{Z}_p^m)^\ell} \Pr\left[\mathbf{G}^{\text{pskp}}_{p,m,k,\tau}(\mathcal{A}, \mathsf{Lk})\right].$$

We state a theorem which captures the *concrete security* of the ADW identification scheme. The theorem streamlines the original analysis of ADW to a precise relation of advantages, which allows us to instantiate parameters of practical sizes.

Theorem 5.1. *Let $\mathcal{G} = (G, G_T, g, \mathbf{e}, p)$ be a group with efficient pairngs. Let $\mathsf{ID} = \mathsf{ADW}_{\mathcal{G},k,m,\tau,r}$ be the ADW identification scheme shown in Fig. 10. Let $\mathcal{A} = (\mathcal{A}.\mathsf{Setup}, \mathcal{A}.\mathsf{Com}, \mathcal{A}.\mathsf{Rsp})$ be a leakage impersonation adversary. Let $q$ denote the number of* H *queries plus the number of* Prover *queries that $\mathcal{A}$.*Setup *and $\mathcal{A}$.*Com *makes. Fig. 14 and Fig. 15 gives two adversaries $\mathcal{A}_{\text{cdh}}$ and $\mathcal{A}_{\text{dl}}$ such that*

$$\mathbf{Adv}^{\text{imp}}_{\mathsf{ID},\ell}(\mathcal{A})^2 \le \mathbf{Adv}^{\text{cdh}}_{\mathcal{G}}(\mathcal{A}_{\text{cdh}}) + m \cdot \mathbf{Adv}^{\text{dl}}_{\mathcal{G}}(\mathcal{A}_{\text{dl}}) \quad (24)$$

$$+ \mathbf{Adv}^{\text{pskp}}_{p,m,k,\tau}(\ell + k/m) + \frac{q}{2^r} + \frac{1}{p}. \quad (25)$$

*Additionally, let $t_1$ be the running time of $\mathcal{A}$.*Setup*, $t_2$ be the running time of $\mathcal{A}$.*Com*, $t_3$ be the running time of $\mathcal{A}$.*Rsp*, and let $t_4$ be the*

| Algorithm ID.Pg() | Algorithm ID.Com(prm) | Algorithm $\mathsf{Derive}^{\mathsf{H}}(R)$ |
|---|---|---|
| For $i \in [m]$ do $g_i \leftarrow\!\!\$\ G$<br>Return $(g_0, \ldots, g_{m-1})$ | $y \leftarrow\!\!\$\ (\mathbb{Z}_p)^m$<br>$a \leftarrow\!\!\$\ \prod_{j=0}^{m-1} g_i^{y[j]}$<br>Return $(a, y)$ | $\mathbf{p} \leftarrow \mathsf{H}(R, [k]^{(\tau)})$<br>$e \leftarrow\!\!\$\ \mathsf{H}(0\|R, [p])$; $c^* \leftarrow\!\!\$\ \mathsf{H}(1\|R, [p])$<br>Return $(\mathbf{p}, e, c^*)$ |
| **Algorithm ID.$\mathsf{Kg}^{\mathsf{H}}$(prm)** | **Algorithm ID.$\mathsf{Rsp}^{\mathsf{H}}$(prm, hlp, sk, st, chl)** | **Algorithm ID.$\mathsf{Vrf}^{\mathsf{H}}$(prm, vk, com, chl, rsp)** |
| $s \leftarrow\!\!\$\ \mathbb{Z}_p$; $\mathsf{vk} \leftarrow g^s$<br>For $i \in [k]$ do<br>  $\mathsf{sk}[i] \leftarrow\!\!\$\ (\mathbb{Z}_p)^m$<br>  $\mathsf{pk}[i] \leftarrow \prod_{j=0}^{m-1} (g_i)^{\mathsf{sk}[i][j]}$<br>  $\sigma[i] \leftarrow\!\!\$\ (\mathsf{H}(i, G)\mathsf{pk}[i])^s$<br>$\mathsf{hlp} \leftarrow (\mathsf{pk}, \sigma)$<br>Return $(\mathsf{sk}, \mathsf{vk}, \mathsf{hlp})$ | $(\mathbf{p}, e, c^*) \leftarrow \mathsf{Derive}^{\mathsf{H}}(\mathsf{chl})$<br>For $j \in [m]$ do<br>  $\mathsf{sk}^*[j] \leftarrow \sum_{i=0}^{\tau-1} (\mathsf{sk}[\mathbf{p}[i]][j]) e^i$<br>$\mathsf{pk}^* \leftarrow \prod_{i=0}^{\tau-1} \mathsf{pk}[\mathbf{p}[i]]^{e^i}$<br>$\sigma^* \leftarrow \prod_{i=0}^{\tau-1} \sigma[\mathbf{p}[i]]^{e^i}$<br>For $j \in [m]$ do $z \leftarrow y[j] + c^* \cdot \mathsf{sk}^*[j]$<br>Return $(\mathsf{pk}^*, \sigma^*, z)$ | $a \leftarrow \mathsf{com}$<br>$(\mathbf{p}, e, c^*) \leftarrow \mathsf{Derive}^{\mathsf{H}}(\mathsf{chl})$<br>$(\mathsf{pk}^*, \sigma^*, z) \leftarrow \mathsf{rsp}$<br>$A \leftarrow (\prod_{i=0}^{m-1} g_i^{z[i]} = a(\mathsf{pk}^*)^{c^*})$<br>$B \leftarrow (\mathbf{e}(\mathsf{pk}^* \prod_{i=0}^{\tau-1} \mathsf{H}(\mathbf{p}[i], G)^{e^i}, \mathsf{vk}) = \mathbf{e}(\sigma^*, g))$<br>Return $(A \wedge B)$ |

**Figure 10: Algorithms of identification scheme** $\mathsf{ID} = \mathsf{ADW}[\mathcal{G}, k, m, \tau, r]$ **associated to bilinear group description** $\mathcal{G} = (G, G_T, g, \mathbf{e}, p)$ **and parameters** $k, m, \tau, r$ **satisfying** $m \geq 2$ **and** $k \geq \tau \geq 1$**. Here** H **is a variable range function, meaning** $\mathsf{H}(\cdot, \mathsf{Rng})$ **returns outputs in the set (described by)** Rng**. In addition, algorithms** Kg, Com, Rsp, Vrf **also takes** prm **as argument.**

| **Computation cost** | | | | | |
|---|---|---|---|---|---|
| | Kg | Com | Chl | Rsp | Vrf |
| Mult $G$ | $k \cdot m$ | $m-1$ | 0 | $2\tau$ | $m+\tau$ |
| Exp $G$ | $k(m+1)+1$ | $m$ | 0 | $2\tau-2$ | $\tau+m+1$ |
| Mult $\mathbb{Z}_p$ | 0 | 0 | 0 | $m$ | 1 |
| Exp $\mathbb{Z}_p$ | $m$ | 0 | 0 | $2\tau$ | $\tau$ |
| $\mathbf{e}$ eval | 0 | 0 | 0 | 0 | 1 |
| **Communication cost** | | | | | |
| $G$ | - | 1 | 0 | 2 | 0 |
| $\mathbb{Z}_p$ | - | 0 | 0 | $m$ | 0 |
| $\{0,1\}^r$ | - | 0 | 1 | 0 | 0 |

**Figure 11: Table illustrating computation and communication cost of different operations of the identification scheme** $\mathsf{ADW}_{\mathcal{G},k,m,\tau,r}$**.** Chl **here represents the challenge phase of the protocol.**

| $m$ | $\tau$ (Us) | $\tau$ (ADW) |
|---|---|---|
| 2 | 718 | 3951 |
| 4 | 349 | 2397 |
| 8 | 245 | 1996 |
| 16 | 201 | 1840 |
| 32 | 180 | 1771 |
| 64 | 169 | 1739 |

**Figure 12: Example parameters for** ADW **scheme to achieve** 128**-bit security. The schemes uses group of size** $p$ **such that** $2^{511} < p < 2^{512}$**, and we impose a bound on the leakage of 10% on a big-key of size** 100 **GB** $= 8 \times 10^{11}$ **bits. For each value of** $m$ **on the left column, we look the value of** $\tau$ **needed to achieve** 128**-bit security for the identification scheme, both using our bound and using ADW's bound.**

*running time of* ID.Kg*. We have that the running time of* $\mathcal{A}_{\mathrm{cdh}}$ *and* $\mathcal{A}_{\mathrm{dl}}$ *is approximately* $t_1 + t_2 + 2 \cdot t_3 + t_4$*.*

The proof of Theorem 5.1 is given in Section 5.1. The following lemma relates $\mathbf{Adv}^{\mathrm{pskp}}_{p,m,k,\tau}(\ell + k/m)$ to the large-alphabet subkey prediction advantage (as bounded in Section 3.3).

Lemma 5.2. *Let* $p, m, k, \tau, \ell$ *be positive integers, then*

$$\mathbf{Adv}^{\mathrm{pskp}}_{p,m,k,\tau}(\ell) \leq \sqrt{\mathbf{Adv}^{\mathrm{skp}}_{pm,k,\tau}(\ell) + \frac{\tau}{p}}.$$

We note that with Lemma 5.2, we can bound the term $\mathbf{Adv}^{\mathrm{pskp}}_{p,k,k,\tau}(\ell)$ for any value $p, m, k, \tau, \ell$. Hence, the only term that is not explicity bounded on the right-hand side of Equation (24) are $\mathbf{Adv}^{\mathrm{cdh}}_{\mathcal{G}}(\mathcal{A})$ and $m \cdot \mathbf{Adv}^{\mathrm{cdh}}_{\mathcal{G}}(\mathcal{A})$, which can be assumed to be small when the CDH and DL problems are suspected to be hard in group $G$.

*Comparison with ADW's analysis.* Our analysis of ADW's identification scheme improves upon the original analysis in the following ways. First, we analyze the scheme in which the challenge is generated using a random oracle directly. (The construction that uses a random oracle to derive the challenge is mentioned to be secure in [2] with no proof.) Second, while ADW's analysis is offered in the asymptotic case, we state and prove a reduction that gives *concrete security*, which lead to practical instantiation of parameters. The reduction gives a bound of the impersonation advantage in terms of three dominating quantities: CDH and DL advantages in $\mathcal{G}$, and a special form of subkey-prediction advantage under polynomial compression, $\mathbf{Adv}^{\mathrm{pskp}}_{p,m,k,\tau}(\mathsf{Lk}, \mathcal{A})$. Hence, giving a good numerical bound of the impersonation advantage amounts to bounding $\mathbf{Adv}^{\mathrm{pskp}}_{p,m,k,\tau}(\mathsf{Lk}, \mathcal{A})$. Here is where we make significant improvements: we use the large-alphabet subkey prediction lemma (Theorem 3.1) as well as a tighter polynomial-evaluation entropy preservation lemma (Lemma 5.2) to give significantly better concrete bounds. The comparison of parameters can be found in Fig. 12.

*Parameter instantiation.* We give an example instantiation of the ADW identification scheme with 128-bits security. First, we find a pairing friendly group $G$ with symmetric pairing $\mathbf{e} : G \times G \to G_T$. Because of the square-root loss of security, we need 256-bit of security for CDH and DL in $G$. Hence, $G$ needs to be of size roughly 512 bits. We consider $\mathcal{G} = (G, G_T, g, \mathbf{e}, p)$, where $p$ is a prime of roughly 512 bits ($2^{511} < p < 2^{512}$). We represent elements in $\mathbb{Z}_p$ using exactly 512 bits. We pick a big key size of 100 GB, i.e. $k^* = 8 \cdot 10^{11}$. For a choice of $m \geq 2$, we have that the block size in bits is $b = m \cdot 512$. We let $k = k^*/b$ be the size of the big key in blocks. We fix a leakage rate of 10%. By Theorem 5.1 and Lemma 5.2, to achieve 128-bit security for the identification scheme, we need 512 bits of security from $\mathbf{Adv}^{\mathrm{skp}}_{pm,k,\tau}(\ell + \frac{k}{m})$. Hence, we need

$$\tau = \mathbf{Probes}_{k^*,\ell^*+k^*/m,s}(m \cdot 512)$$

probes. Values of $\mathbf{Probes}_{k^*,\ell^*+k^*/m,s}(m \cdot 512)$ versus various values of $m$ is shown in Fig. 12 using both our bound and ADW's bound.

*Entropy preservation under polynomial evaluation.* Lemma 5.2 relates the prediction advantage to the large-alphabet subkey prediction advantage. Note that our bound is quantitatively better than [2, Corollary A.1]. In particular, we prove $\frac{1}{2}$ rate entropy preservation while ADW proves a rate of $\frac{1}{3}$. Before proving the lemma, we define the following quantities for jointly distributed random variables $(X, Y)$. Let $X$ be a random variable, the *prediction* and *collision* probability of $X$ is defined, respectively, to be

$$\mathsf{Pred}(X) = \max_x \Pr[X = x], \quad \mathsf{CP}(X) = \Pr[X = X'],$$

where $X'$ is an independent random variable that is identically distributed to $X$. Additionally, suppose that $(X, Y)$ are jointly distributed, we define the *conditional* prediction and collision probability of $X$ *given* $Y$, respectively, to be

$$\widetilde{\mathsf{Pred}}(X \mid Y) = \mathbf{E}_Y[\mathsf{Pred}(X \mid Y)],$$
$$\widetilde{\mathsf{CP}}(X \mid Y) = \mathbf{E}_Y[\mathsf{CP}(X \mid Y)].$$

We note that $\mathsf{Pred}(X \mid Y)$ and $\mathsf{CP}(X \mid Y)$ are random variables in $Y$. We need the following well-known lemma,

Lemma 5.3. *Let $(X, Y)$ be jointly distributed random variables, then*

$$\widetilde{\mathsf{CP}}(X \mid Y) \leq \widetilde{\mathsf{Pred}}(X \mid Y) \leq \sqrt{\widetilde{\mathsf{CP}}(X \mid Y)}.$$

The proof of Lemma 5.3 is given in the full version [4].

Proof (of Lemma 5.2). Let $\mathcal{A}$ be any adversary and $\mathsf{Lk} : [q]^k \to [q]^\ell$ be a leakage function. Consider the sample space defined by the experiment $\mathbf{G}^{\mathrm{pskp}}_{p,m,k,\tau}(\mathcal{A}, \mathsf{Lk})$ (all the coins used by the experiment and adversary $\mathcal{A}$). We consider all the variables used inside $\mathbf{G}^{\mathrm{pskp}}_{p,m,k,\tau}(\mathcal{A}, \mathsf{Lk})$, e.g. $\mathsf{sk}^*, L = \mathsf{Lk}(\boldsymbol{K})$, as random variables. We note that

$$\Pr\left[\mathbf{G}^{\mathrm{pskp}}_{p,m,k,\tau}(\mathcal{A}, \mathsf{Lk})\right] \leq \widetilde{\mathsf{Pred}}(\mathsf{sk}^* \mid \mathbf{p}, L, e).$$

Furthermore, by Lemma 5.3,

$$\widetilde{\mathsf{Pred}}(\mathsf{sk}^* \mid \mathbf{p}, L, e) \leq \sqrt{\widetilde{\mathsf{CP}}(\mathsf{sk}^* \mid \mathbf{p}, L, e)}.$$

We now need to bound $\widetilde{\mathsf{CP}}(\mathsf{sk}^* | \mathbf{p}, L, e)$. To compute this quantity. We consider another independent execution of $\mathbf{G}^{\mathrm{pskp}}_{p,m,k,\tau}(\mathcal{A}, \mathsf{Lk})$, where the variables in the second execution is denoted with $'$, e.g. $\mathsf{sk}'$. We restrict to the event that $\mathsf{Lk}(\mathsf{sk}) = \mathsf{Lk}(\mathsf{sk}')$ and $\mathbf{p} = \mathbf{p}'$. We define polynomials $p_1, \ldots, p_j$, which are functions of $\mathsf{sk}, \mathsf{sk}', \mathbf{p}$, $p_j(x) = \sum_{i=0}^{\tau-1}(\mathsf{sk}[\mathbf{p}[i]][j] - \mathsf{sk}'[\mathbf{p}[i]][j])x^i$. Notice that these polynomials are of degree at most $\tau$. If $\mathsf{sk} \neq \mathsf{sk}'$, then at least one of $p_j$ is a non-zero polynomial, and has at most $\tau$ roots. Hence, if $\mathsf{sk} \neq \mathsf{sk}'$, over a independently uniform $e$, the probability that $p_j(e) = 0$ is at most $\frac{\tau}{p}$ when $p_j$ is not the zero polynomial. Finally, we derive that

$$\begin{aligned}
&\widetilde{\mathsf{CP}}(\mathsf{sk}^* \mid \mathbf{p}, L, e) \\
&= \mathbf{E}_{\mathbf{p},L,e}\left[\mathsf{CP}(\mathsf{sk}^* \mid \mathbf{p}, L, e)\right] \\
&\leq \mathbf{E}_{\mathbf{p},L,e}\Big[\Pr\left[\mathsf{sk}[\mathbf{p}] = \mathsf{sk}'[\mathbf{p}] \mid \mathbf{p}, L, e\right] \\
&\qquad + \Pr\left[\forall j \in [m] : p_j(e) = 0 \mid \mathsf{sk}[\mathbf{p}] \neq \mathsf{sk}'[\mathbf{p}], \mathbf{p}, L, e\right]\Big] \\
&= \widetilde{\mathsf{CP}}(\mathsf{sk}[\mathbf{p}] \mid \mathbf{p}, L) + \mathbf{E}_e\left[\Pr\left[\forall j \in [m] : p_j(e) = 0\right]\right] \\
&\leq \widetilde{\mathsf{Pred}}(\mathsf{sk}[\mathbf{p}] \mid \mathbf{p}, L) + \frac{\tau}{p} \\
&\leq \mathbf{Adv}^{\mathrm{skp}}_{pm,k,\tau}(\ell) + \frac{\tau}{p}.
\end{aligned}$$

□

## 5.1 Proof of Theorem 5.1

We follow the proof technique used by [2].

Proof. (of Theorem 5.1). Let $\mathsf{ID} = \mathsf{ADW}_{\mathcal{G},k,m,\tau,r}$ be the ADW identification scheme. The reduction is very similar to the reduction from [2, Appendix B.5]. Rewind attemps to run a given leakage impersonation adverasry $\mathcal{A}$ twice with two different programmed challenges that only differ in the element $c^*$ ($R$ and $e$ stay the same). Rewind takes an algorithm Gen that generates $(\mathsf{prm}, \mathsf{vk}, \mathsf{sk}, \mathsf{hlp}, T)$, where $T$ is the table used by H. Rewind simulates H for $\mathcal{A}$ using $\mathsf{H}_{\mathsf{Rewind}}$ as decribed by the code. Rewind returns the success status of the rewinding process, along with the two responses of the two executions $(\mathsf{rsp}_1, \mathsf{rsp}_2)$, plus the honest response ($\mathsf{rsp}^*$) and the honestly generated and compressed secrete key ($\mathsf{sk}^*$). Let $x \in \{1, 2\}$, we use $\Pr[\mathsf{Rewind}^x(\mathsf{Gen}, \mathcal{A})]$ to denote the probablity that the first component of the output of $\mathsf{Rewind}^x$ is true. First, using the well-known rewind technique [6], we will argue that

$$\Pr[\mathsf{Rewind}^1(\mathsf{Gen}, \mathcal{A})] \geq \mathbf{Adv}^{\mathrm{imp}}_{\mathsf{ID},\ell}(\mathcal{A})^2 - \frac{1}{p}. \tag{26}$$

We now justify Equation (26). We consider the event that the flags $A, B, C$ are all set to true. Notice that the marginal probability that $A$ is true and the marginal probability that $B$ is ture are both exactly $\Pr[\mathbf{G}^{\mathrm{imp}}_{\mathsf{ID},\ell}(\mathcal{A})]$. We partition the random tape for $\mathbf{G}^{\mathrm{imp}}_{\mathsf{ID},\ell}(\mathcal{A})$ into two parts: the random tape that is used upto right before $\mathcal{A}$.Rsp is run, and the rest of the tape that is used after $\mathcal{A}$.Rsp starts its execution. Let $T$ be a random variable denoting the first part of the random tape. For any value of $T$, say $t$, we let $\mathbf{G}(t)$ be the game $\mathbf{G}^{\mathrm{imp}}_{\mathsf{ID},\ell}(\mathcal{A})$

Game $\mathrm{Rewind}^1(\mathrm{Gen}, \mathcal{A})$ $\boxed{\mathrm{Rewind}^2(\mathrm{Gen}, \mathcal{A})}$

$(\mathsf{prm}, \mathsf{vk}, \mathsf{sk}, \mathsf{hlp}, T) \leftarrow \mathrm{Gen}(); s \leftarrow 0$
$\mathsf{st} \leftarrow\$ \mathcal{A}.\mathrm{Setup}^{\mathrm{Leak}, \mathrm{Prover}, \mathsf{H}_{\mathrm{Rewind}}}(\mathsf{prm}, \mathsf{vk}, \mathsf{hlp})$
$(\mathsf{com}, \mathsf{st}') \leftarrow \mathcal{A}.\mathrm{Com}^{\mathsf{H}_{\mathrm{Rewind}}}(\mathsf{st})$; $\mathsf{chl} \leftarrow \{0,1\}^{\mathrm{ID.Chl}}$
$T_1[0\|\mathsf{chl}, [p]] \leftarrow\$ [p]$; $T_1[1\|\mathsf{chl}, [p]] \leftarrow \perp$
$T_1[\mathsf{chl}, [k]^{(\tau)}] \leftarrow\$ [k]^{(\tau)}$; $T_2 \leftarrow T_1$
If $C[\mathsf{chl}]$ then
  $\mathsf{bad} \leftarrow \mathsf{true}$
  $\boxed{T[0\|\mathsf{chl}, [p]] \leftarrow \perp; T[1\|\mathsf{chl}, [p]] \leftarrow \perp; T[\mathsf{chl}, [k]^{(\tau)}] \leftarrow \perp}$
$\mathsf{rsp}_1 \leftarrow \mathcal{A}.\mathrm{Rsp}^{\mathsf{H}_{\mathrm{Rewind}}[T_1]}(\mathsf{st}', \mathsf{chl})$
$\mathsf{rsp}_2 \leftarrow \mathcal{A}.\mathrm{Rsp}^{\mathsf{H}_{\mathrm{Rewind}}[T_2]}(\mathsf{st}', \mathsf{chl})$
For $j \in [m]$ do $y[j] \leftarrow 0$
$(\mathsf{pk}^*, \sigma^*, z^*) \leftarrow \mathrm{ADW.Rsp}^{\mathsf{H}_{\mathrm{Rewind}}[T_2]}(\mathsf{prm}, \mathsf{hlp}, \mathsf{vk}, \mathsf{sk}, y, \mathsf{chl})$
$(\mathbf{p}, c, e) \leftarrow \mathrm{Derive}^{\mathsf{H}}(\mathsf{chl})$
For $j \in [m]$ do $\mathsf{sk}^*[j] \leftarrow \sum_{i=0}^{\tau-1} (\mathsf{sk}[\mathbf{p}[i]][j]) e^i$
$A \leftarrow \mathrm{Vrf}^{\mathsf{H}_{\mathrm{Rewind}}[T_1]}(\mathsf{prm}, \mathsf{vk}, \mathsf{com}, \mathsf{chl}, \mathsf{rsp}_1)$
$B \leftarrow \mathrm{Vrf}^{\mathsf{H}_{\mathrm{Rewind}}[T_2]}(\mathsf{prm}, \mathsf{vk}, \mathsf{com}, \mathsf{chl}, \mathsf{rsp}_2)$
$C \leftarrow (T_1[1\|\mathsf{chl}, [p]] \neq T_2[1\|\mathsf{chl}, [p]])$
Return $(A \wedge B \wedge C, \mathsf{rsp}_1, \mathsf{rsp}_2, \mathsf{rsp}^*, \mathsf{sk}^*)$

$\mathsf{H}_{\mathrm{Rewind}}[T'](x, \mathrm{Rng})$

If $\mathrm{Rng} = [p]$ then $b\|x \leftarrow x$
$C[x] \leftarrow \mathsf{true}$
If $T[x, \mathrm{Rng}]$ then return $T[x, \mathrm{Rng}]$
If $T'$ then
  If not $T'[x, \mathrm{Rng}]$ then $T'[x, \mathrm{Rng}] \leftarrow\$ \mathrm{Rng}$
  Return $T'[x, \mathrm{Rng}]$
Else if not $T[x, \mathrm{Rng}]$ then $T[x, \mathrm{Rng}] \leftarrow\$ \mathrm{Rng}$
Return $T[x, \mathrm{Rng}]$

Gen()

$\mathsf{prm} \leftarrow\$ \mathrm{Pg}()$; $(\mathsf{vk}, \mathsf{sk}, \mathsf{hlp}) \leftarrow\$ \mathrm{Kg}(\mathsf{prm})$
Return $(\mathsf{prm}, \mathsf{vk}, \mathsf{sk}, \mathsf{hlp}, \perp)$

Prover($i$, args)

If $\mathsf{pst}[i] = \perp$ then // Commit
  $(\mathsf{pcom}[i], \mathsf{pst}[i]) \leftarrow\$ \mathrm{ID.Com}(\mathsf{prm})$
  Return $\mathsf{pcom}[i]$
Else If $\mathsf{prsp}[i] = \perp$ then // Response
  $\mathsf{prsp}[i] \leftarrow\$ \mathrm{ID.Rsp}^{\mathsf{H}_{\mathrm{Rewind}}}(\mathsf{prm}, \mathsf{hlp}, \mathsf{sk}, \mathsf{pst}[i], \mathrm{args})$
  Return $\mathsf{prsp}[i]$
Return $\perp$

**Figure 13: Game $\mathrm{Rewind}^1$ and $\mathrm{Rewind}^2$ (boxed). The oracle Leak is the same as the one given in Fig. 7.**

Adversary $\mathcal{A}_{\mathrm{cdh}}(\mathcal{G}, v, h)$

$(G, G_T, g, \mathbf{e}, p) \leftarrow \mathcal{G}$
$(t, \mathsf{rsp}_1, \mathsf{rsp}_2, \mathsf{rsp}^*, \mathsf{sk}^*) \leftarrow \mathrm{Rewind}^2(\mathrm{Gen}_{\mathrm{cdh}}, \mathcal{A})$
$(\mathsf{pk}_1^*, \sigma_1^*, z^{(1)}) \leftarrow \mathsf{rsp}_1$
$(\mathsf{pk}_2^*, \sigma_2^*, z^{(2)}) \leftarrow \mathsf{rsp}_2$
$(\mathsf{pk}^*, \sigma^*, z^*) \leftarrow \mathsf{rsp}^*$
$\hat{\sigma} \leftarrow ((\sigma_1^*)^{c_1^*}/(\sigma_2^*)^{c_2^*}) \cdot {\sigma^*}^{c_2^* - c_1^*}$
$\omega = \sum_{j=1}^{m} \gamma_j (z_j^{(1)} - z_j^{(2)} - x_j^*(c_1^* - c_2^*))$
$s' \leftarrow (\hat{\sigma})^{1/\omega}$
Return $s'$

$\mathrm{Gen}_{\mathrm{cdh}}()$

$(G, G_T, g, \mathbf{e}, p) \leftarrow \mathcal{G}$
For $j \in [m]$ do
  $\gamma_j \leftarrow \mathbb{Z}_p$; $g_j \leftarrow h^{\gamma_j}$
$\mathsf{prm} = (g_0, \ldots, g_{m-1}, g)$; $\mathsf{vk} \leftarrow v$
For $i \in [m]$ do
  $\mathsf{sk}[i] \leftarrow [p]^m$; $\mathsf{pk}[i] \leftarrow \prod_{j=0}^{m-1} g_j^{\mathsf{sk}[i][j]}$
  $\beta_i \leftarrow [p]$; $\sigma[i] \leftarrow \mathsf{vk}^{\beta_i}$
  $T[i, G] \leftarrow g^{\beta_i}/\mathsf{pk}[i]$
$\mathsf{hlp} \leftarrow (\mathsf{pk}, \sigma)$
Return $(\mathsf{prm}, \mathsf{vk}, \mathsf{sk}, \mathsf{hlp}, T)$

**Figure 14: Adversary $\mathcal{A}_{\mathrm{cdh}}$.**

Adversary $\mathcal{A}_{\mathrm{dl}}(\mathcal{G}, X)$

$(t, \mathsf{rsp}_1, \mathsf{rsp}_2, \mathsf{rsp}^*, \mathsf{sk}^*) \leftarrow \mathrm{Rewind}^2(\mathrm{Gen}_{\mathrm{dl}}, \mathcal{A})$
$(\mathsf{pk}_1^*, \sigma_1^*, z^{(1)}) \leftarrow \mathsf{rsp}_1$
$(\mathsf{pk}_2^*, \sigma_2^*, z^{(2)}) \leftarrow \mathsf{rsp}_2$
$(\mathsf{pk}^*, \sigma^*, z^*) \leftarrow \mathsf{rsp}^*$
For $j = 1, \ldots, m$ do
  $\hat{\mathsf{sk}}^*[j] \leftarrow (z^{(1)}[j] - z^{(2)}[j])/(c_1^* - c_2^*)$
$x \leftarrow (\sum_{i \in [m]-\{\rho\}} x_i(\hat{\mathsf{sk}}^*[i] - \mathsf{sk}^*[i]))/(\mathsf{sk}^*[\rho] - \hat{\mathsf{sk}}^*[\rho])$
Return $x$

$\mathrm{Gen}_{\mathrm{dl}}()$

$\rho \leftarrow\$ [m]$; $g_\rho \leftarrow X$
For $j \in [m] - \{\rho\}$ do
  $x_j \leftarrow\$ \mathbb{Z}_p$; $g_j \leftarrow g^{x_j}$
$\mathsf{prm} = (g_0, \ldots, g_{m-1}, g)$
$(\mathsf{pk}, \mathsf{sk}, \mathsf{hlp}) \leftarrow \mathrm{ADW.Kg}(\mathsf{prm})$
Return $(\mathsf{prm}, \mathsf{pk}, \mathsf{sk}, \mathsf{hlp}, \perp)$

**Figure 15: Adversary $\mathcal{A}_{\mathrm{dl}}$.**

with the first part of random tape fixed to $t$. We have that

$$\begin{aligned}
\Pr\left[\mathrm{Rewind}^1(\mathrm{Gen}, \mathcal{A})\right] &= \Pr[A \wedge B \wedge C] \\
&= \mathbf{E}_T\left[\Pr[A \wedge B \wedge C \mid T]\right] \\
&\geq \mathbf{E}_T\left[\Pr[A \wedge B \mid T] - \Pr[\neg C \mid T]\right] \\
&= \mathbf{E}_T[\Pr[\mathrm{G}(T)]^2] - \frac{1}{p} \\
&\geq \Pr\left[\mathrm{G}^{\mathrm{imp}}_{\mathrm{ID},\ell}(\mathcal{A})\right]^2 - \frac{1}{p}\ ,
\end{aligned}$$

where at the last step we used Jensen's inequality and the convexity of squaring. This justifies Equation (26). Second, we argue that

$$\begin{aligned}
&\Pr[\mathrm{Rewind}^1(\mathrm{Gen}, \mathcal{A})] - \Pr[\mathrm{Rewind}^2(\mathrm{Gen}, \mathcal{A})] \\
&\leq \Pr[\mathrm{Rewind}^1(\mathrm{Gen}, \mathcal{A}) \text{ sets bad}] = \frac{q}{2^r}.
\end{aligned} \tag{27}$$

This is because the size of the table $C$ is upper-bounded by the number of queries that $\mathcal{A}$.Setup and $\mathcal{A}$.Com makes to H and Derive, which is $q$. Next, we attempt to bound $\Pr[\mathrm{Rewind}^2(\mathrm{Gen}, \mathcal{A})]$. We

define the following events in the game $\mathsf{Rewind}^2(\mathsf{Gen}, \mathcal{A})$.

$$E: \quad A \wedge B \wedge C \wedge \left(\frac{(\mathsf{pk}_1^*)^{(c_1^*)}}{(\mathsf{pk}_2^*)^{(c_2^*)}} = (\mathsf{pk}^*)^{c_1^*-c_2^*}\right),$$

$$\overline{E}: \quad A \wedge B \wedge C \wedge \left(\frac{(\mathsf{pk}_1^*)^{(c_1^*)}}{(\mathsf{pk}_2^*)^{(c_2^*)}} \neq (\mathsf{pk}^*)^{c_1^*-c_2^*}\right).$$

Notice that per construction of the events,

$$\Pr[\mathsf{Rewind}^2(\mathsf{Gen}, \mathcal{A})] = \Pr[E] + \Pr[\overline{E}]. \tag{28}$$

Consider $\mathcal{A}_{\mathrm{cdh}}$ (Fig. 14) and $\mathcal{A}_{\mathrm{dl}}$ (Fig. 15), which attemps to break CDH and DL problems, respectively, using $\mathsf{Rewind}^2$. We will show the following (in)equalities

$$\Pr[\overline{E}] = \mathbf{Adv}^{\mathrm{cdh}}_{\mathcal{G}}(\mathcal{A}_{\mathrm{cdh}}), \tag{29}$$

and

$$\Pr[E] \leq m \cdot \mathbf{Adv}^{\mathrm{dl}}_{\mathcal{G}}(\mathcal{A}_{\mathrm{dl}}) + \mathbf{Adv}^{\mathrm{pskp}}_{p,m,k,\tau}(\ell + \frac{k}{m}). \tag{30}$$

This part of the analysis follows from [2, Appendix B.5] and we restate their derivation here. Assume $E$ or $\overline{E}$, since the signatures verifies, for $w = \prod_{i=0}^{\tau} \mathsf{H}(\mathsf{p}[i], G)^{e^i}$, we have

$$\sigma^* = (\mathsf{pk}^* w)^s \ , \ \sigma_1^* = (\mathsf{pk}_1^* w)^s \ , \ \sigma_2^* = (\mathsf{pk}_2^* w)^s .$$

If $\overline{E}$, the following two values are distinct

$$\frac{(\sigma_1)^{c_1^*}}{(\sigma_2)^{c_2^*}} = \left(w^{(c_1^*-c_2^*)} \frac{(\mathsf{pk}_1^*)^{c_1^*}}{(\mathsf{pk}_2^*)^{c_2^*}}\right)^s \ , \ (\sigma^*)^{c_1^*-c_2^*} = \left(w^{c_1^*-c_2^*} (\mathsf{pk})^{c_1^*-c_2^*}\right)^s .$$

Hence, the value $\hat{\sigma}$ computed by $\mathcal{A}_{\mathrm{cdh}}$ is

$$\hat{\sigma} = \left(\frac{(\mathsf{pk}_1^*)^{c_1^*}}{(\mathsf{pk}_2^*)^{c_2^*}} \cdot \frac{1}{(\mathsf{pk}^*)^{c_1^*-c_2^*}}\right)^s = \left(g^{\omega}\right)^s .$$

Therefore, $\mathcal{A}_{\mathrm{cdh}}$ can compute $g^s$ and solve the CDH problem that it was given. This concludes the proof for Equation (29). If $E$, then we claim that $\mathsf{sk}^* = \hat{\mathsf{sk}}$ with probability at most $\mathbf{Adv}^{\mathrm{pskp}}_{p,k,k,\tau}(\ell + k/m)$. This is true per definition of $\mathbf{Adv}^{\mathrm{pskp}}_{p,k,k,\tau}(\ell + k/m)$. Notice that if $\mathsf{sk}^* \neq \hat{\mathsf{sk}}$, then with probability $\frac{1}{m}$, $\mathcal{A}_{\mathrm{dl}}$ can solved the DL problem that it embedded into the parameters. This is because $\mathcal{A}_{\mathrm{dl}}$ has two representation of $\mathsf{pk}^*$ in the basis $g_0, \ldots, g_{m-1}$, namely $\mathsf{sk}^*$ and $\hat{\mathsf{sk}}$. This concludes the proof of Equation (30). Notice that Equations (26), (28), (27), (29), and (30) together implies the theorem. Finally, notice that $\mathcal{A}_{\mathrm{cdh}}$ and $\mathcal{A}_{\mathrm{dl}}$ has roughly the running time of $\mathsf{Rewind}^2$ and ADW.Kg, which is about $t_1 + t_2 + 2t_3 + t_4$. □

## 6 CONCLUSION

Towards the goal of defending against key exfiltration, we have made practical improvements for big-key cryptography in the symmetric encryption and public-key identification setting. We have identified the *probe complexity* of big-key access as the dominant cost in both settings. We have significantly improved the probe complexity of existing schemes via our large-alphabet subkey prediction lemma, which is our main technical contribution. Improving upon BKR's results, our big-key symmetric encryption scheme uses less probes by utilizes blocks of the underlying storage medium (instead of bits). Improving upon ADW's results and analysis, we prove a reduction that gives *concrete security*, which we use to find instantiation of parameters for the big-key identification protocol that is significantly more efficient. We believe that our new large-alphabet subkey prediction lemma is key to bridging the gap between theory and practice for the bounded-retrieval model.

For future research, we point out that the reduction in proof of Theorem 5.1 incurs two square-root losses. Improving either will further increase the efficiency of the identification scheme.

## 7 ACKNOWLEDGEMENTS

We thank the anonymous reviewers for their helpful comments. We thank Yevgeniy Dodis and Daniel Wichs for the helpful discussions.

## REFERENCES

[1] Joël Alwen, Yevgeniy Dodis, Moni Naor, Gil Segev, Shabsi Walfish, and Daniel Wichs. 2010. Public-Key Encryption in the Bounded-Retrieval Model. In *EUROCRYPT 2010 (LNCS)*, Henri Gilbert (Ed.), Vol. 6110. Springer, Heidelberg, 113–134.

[2] Joël Alwen, Yevgeniy Dodis, and Daniel Wichs. 2009. Leakage-Resilient Public-Key Cryptography in the Bounded-Retrieval Model. In *CRYPTO 2009 (LNCS)*, Shai Halevi (Ed.), Vol. 5677. Springer, Heidelberg, 36–54.

[3] Joël Alwen, Yevgeniy Dodis, and Daniel Wichs. 2009. Leakage-Resilient Public-Key Cryptography in the Bounded-Retrieval Model. Cryptology ePrint Archive, Report 2009/160. (2009). http://eprint.iacr.org/2009/160.

[4] Mihir Bellare and Wei Dai. 2017. Defending Against Key Exfiltration: Efficiency Improvements for Big-Key Cryptography via Large-Alphabet Subkey Prediction. Cryptology ePrint Archive, 2017. (2017).

[5] Mihir Bellare, Daniel Kane, and Phillip Rogaway. 2016. Big-Key Symmetric Encryption: Resisting Key Exfiltration. In *CRYPTO 2016, Part I (LNCS)*, Matthew Robshaw and Jonathan Katz (Eds.), Vol. 9814. Springer, Heidelberg, 373–402. https://doi.org/10.1007/978-3-662-53018-4_14

[6] Mihir Bellare and Adriana Palacio. 2002. GQ and Schnorr Identification Schemes: Proofs of Security against Impersonation under Active and Concurrent Attacks. In *CRYPTO 2002 (LNCS)*, Moti Yung (Ed.), Vol. 2442. Springer, Heidelberg, 162–177.

[7] Mihir Bellare, Bertram Poettering, and Douglas Stebila. 2016. From Identification to Signatures, Tightly: A Framework and Generic Transforms. In *ASIACRYPT 2016, Part II (LNCS)*, Jung Hee Cheon and Tsuyoshi Takagi (Eds.), Vol. 10032. Springer, Heidelberg, 435–464. https://doi.org/10.1007/978-3-662-53890-6_15

[8] Mihir Bellare and Phillip Rogaway. 2006. The Security of Triple Encryption and a Framework for Code-Based Game-Playing Proofs. In *EUROCRYPT 2006 (LNCS)*, Serge Vaudenay (Ed.), Vol. 4004. Springer, Heidelberg, 409–426.

[9] Dan Boneh, Ben Lynn, and Hovav Shacham. 2001. Short Signatures from the Weil Pairing. In *ASIACRYPT 2001 (LNCS)*, Colin Boyd (Ed.), Vol. 2248. Springer, Heidelberg, 514–532.

[10] David Cash, Yan Zong Ding, Yevgeniy Dodis, Wenke Lee, Richard J. Lipton, and Shabsi Walfish. 2007. Intrusion-Resilient Key Exchange in the Bounded Retrieval Model. In *TCC 2007 (LNCS)*, Salil P. Vadhan (Ed.), Vol. 4392. Springer, Heidelberg, 479–498.

[11] Feng Chen, David A Koufaty, and Xiaodong Zhang. 2009. Understanding intrinsic characteristics and system implications of flash memory based solid state drives. In *ACM SIGMETRICS Performance Evaluation Review*, Vol. 37. ACM, 181–192.

[12] Crypto++ 5.6.5 Benchmarks 2015. https://www.cryptopp.com/benchmarks.html. (2015). Accessed: 2017-05-18.

[13] Giovanni Di Crescenzo, Richard J. Lipton, and Shabsi Walfish. 2006. Perfectly Secure Password Protocols in the Bounded Retrieval Model. In *TCC 2006 (LNCS)*, Shai Halevi and Tal Rabin (Eds.), Vol. 3876. Springer, Heidelberg, 225–244.

[14] Stefan Dziembowski. 2006. Intrusion-Resilience Via the Bounded-Storage Model. In *TCC 2006 (LNCS)*, Shai Halevi and Tal Rabin (Eds.), Vol. 3876. Springer, Heidelberg, 207–224.

[15] Amos Fiat and Adi Shamir. 1987. How to Prove Yourself: Practical Solutions to Identification and Signature Problems. In *CRYPTO'86 (LNCS)*, Andrew M. Odlyzko (Ed.), Vol. 263. Springer, Heidelberg, 186–194.

[16] McAfee. 2015. Stop data exfiltration. (2015). August 2015. https://www.mcafee.com/us/resources/solution-briefs/sb-quarterly-threats-aug-2015-1.pdf.

[17] Tatsuaki Okamoto. 1993. Provably Secure and Practical Identification Schemes and Corresponding Signature Schemes. In *CRYPTO'92 (LNCS)*, Ernest F. Brickell (Ed.), Vol. 740. Springer, Heidelberg, 31–53.

[18] Herbert Robbins. 1955. A remark on Stirling's formula. *The American Mathematical Monthly* 62, 1 (1955), 26–29.

[19] Claus-Peter Schnorr. 1991. Efficient Signature Generation by Smart Cards. *Journal of Cryptology* 4, 3 (1991), 161–174.