Order-Preserving Symmetric Encryption

Alexandra Boldyreva, Nathan Chenette, Younho Lee and Adam O'Neill

Georgia Institute of Technology, Atlanta, GA, USA
{sasha,nchenette}@gatech.edu, {younho,amoneill}@cc.gatech.edu

Abstract

We initiate the cryptographic study of order-preserving symmetric encryption (OPE), a primitive suggested in the database community by Agrawal et al. (SIGMOD '04) for allowing efficient range queries on encrypted data. Interestingly, we first show that a straightforward relaxation of standard security notions for encryption such as indistinguishability against chosen-plaintext attack (IND-CPA) is unachievable by a practical OPE scheme. Instead, we propose a security notion in the spirit of pseudorandom functions (PRFs) and related primitives asking that an OPE scheme look "as-random-as-possible" subject to the order-preserving constraint. We then design an efficient OPE scheme and prove its security under our notion based on pseudorandomness of an underlying blockcipher. Our construction is based on a natural relation we uncover between a random order-preserving function and the hypergeometric probability distribution. In particular, it makes black-box use of an efficient sampling algorithm for the latter.

1 Introduction

MOTIVATION. Order-preserving symmetric encryption (OPE) is a deterministic encryption scheme (aka. cipher) whose encryption function preserves numerical ordering of the plaintexts. OPE has a long history in the form of one-part codes, which are lists of plaintexts and the corresponding ciphertexts, both arranged in alphabetical or numerical order so only a single copy is required for efficient encryption and decryption. One-part codes were used, for example, during World War I [3]. A more formal treatment of the concept of order-preserving symmetric encryption (OPE) was proposed in the database community by Agrawal et al. [1]. The reason for new interest in such schemes is that they allow efficient range queries on encrypted data. That is, a remote untrusted database server is able to index the (sensitive) data it receives, in encrypted form, in a data structure that permits efficient range queries (asking the server to return ciphertexts in the database whose decryptions fall within a given range, say $[a, b]$). By "efficient" we mean in time logarithmic (or at least sub-linear) in the size of the database, as performing linear work on each query is prohibitively slow in practice for large databases.

In fact, OPE not only allows efficient range queries, but allows indexing and query processing to be done exactly and as efficiently as for unencrypted data, since a query just consists of the encryptions of a and b and the server can locate the desired ciphertexts in logarithmic-time via standard tree-based data structures. Indeed, subsequent to its publication, [1] has been referenced widely in the database community, and OPE has also been suggested for use in in-network aggregation on encrypted data

in sensor networks [30] and as a tool for applying signal processing techniques to multimedia content protection [13]. Yet a cryptographic study of OPE in the provable-security tradition never appeared. Our work aims to begin to remedy this situation.

Related Work. Our work extends a recent line of research in the cryptographic community addressing efficient (sub-linear time) search on encrypted data, which has been addressed by [2] in the symmetric-key setting and [6, 10, 7] in the public-key setting. However, these works focus mainly on simple exact-match queries. Development and analysis of schemes allowing more complex query types that are used in practice (e.g. range queries) has remained open.

The work of [24] suggested enabling efficient range queries on encrypted data not by using OPE but so-called *prefix-preserving encryption* (PPE) [31, 5]. Unfortunately, as discussed in [24, 2], PPE schemes are subject to certain attacks in this context; particular queries can completely reveal some of the underlying plaintexts in the database. Moreover, their use necessitates specialized data structures and query formats, which practitioners would prefer to avoid.

Allowing range queries on encrypted data in the public-key setting was studied in [11, 28]. While their schemes provably provide strong security, they are not efficient in our setting, requiring to scan the whole database on every query.

Finally, we clarify that [1], in addition to suggesting the OPE primitive, *does* provide a construction. However, the construction is rather ad-hoc and has certain limitations, namely its encryption algorithm must take as input all the plaintexts in the database. It is not always practical to assume that users know all these plaintexts in advance, so a stateless scheme whose encryption algorithm can process single plaintexts on the fly is preferable. Moreover, [1] does not define security nor provide any formal security analysis.

Defining security of OPE. Our first goal is to devise a rigorous definition of security that OPE schemes should satisfy. Of course, such schemes cannot satisfy all the standard notions of security, such as indistinguishability against chosen-plaintext attack (IND-CPA), as they are not only deterministic, but also leak the order-relations among the plaintexts. So, although we cannot target for the strongest security level, we want to define the best possible security under the order-preserving constraint that the target-applications require. (Such an approach was taken previously in the case of deterministic public-key encryption [6, 10, 7], on-line ciphers [5], and deterministic authenticated encryption [27].)

Weakening IND-CPA. One approach is to try to weaken the IND-CPA definition appropriately. Indeed, in the case of deterministic symmetric encryption this was done by [8], which formalizes a notion called *indistinguishability under distinct chosen-plaintext attack* or IND-DCPA. (The notion was subsequently applied to MACs in [4].) Since deterministic encryption leaks equality of plaintexts, they restrict the adversary in the IND-CPA experiment to make queries to its left-right-encryption-oracle of the form $(x_0^1, x_1^1), \ldots, (x_0^q, x_1^q)$ such that $x_0^1, \ldots, x_0^q$ are all distinct and $x_1^1, \ldots, x_1^q$ are all distinct. We generalize this to a notion we call *indistinguishability under ordered chosen-plaintext attack* or IND-OCPA, asking these sequences instead to satisfy the same *order relations*. (See Section 3.2.) Surprisingly, we go on to show that this plausible-looking definition is not very useful for us, because it cannot be achieved by an OPE scheme unless the size of its ciphertext-space is exponential in the size of its plaintext-space.

An alternative approach. Instead of trying to further restrict the adversary in the IND-OCPA definition, we turn to an approach along the lines of pseudorandom functions (PRFs) or permutations (PRPs), requiring that no adversary can distinguish between oracle access to the encryption algorithm of the scheme or a corresponding "ideal" object. In our case the latter is a random order-preserving

function with the same domain and range. Since order-preserving functions are injective, it also makes sense to aim for a stronger security notion that additionally gives the adversary oracle access to the decryption algorithm or the inverse function, respectively. We call the resulting notion POPF-CCA for *pseudorandom order-preserving function against chosen-ciphertext attack*.

Towards a construction. After having settled on the POPF-CCA notion, we would naturally like to construct an OPE scheme meeting it. Essentially, the encryption algorithm of such a scheme should behave similarly to an algorithm that samples a random order-preserving function from a specified domain and range on-the-fly (dynamically as new queries are made). (Here we note a connection to implementing huge random objects [18] and lazy sampling [9].) But it is not immediately clear how this can be done; blockciphers, our usual tool in the symmetric-key setting, do not seem helpful in preserving plaintext order. Our construction takes a different route, borrowing some tools from probability theory. We first uncover a relation between a random order-preserving function and the hypergeometric (HG) and negative hypergeometric (NHG) probability distributions.

The connection to NHG. To gain some intuition, first observe that any order-preserving function f from $\{1, \ldots, M\}$ to $\{1, \ldots, N\}$ can be uniquely represented by a combination of M out of N ordered items (see Proposition 4.1). Now let us recall a probability distribution that deals with selections of such combinations. Imagine we have N balls in a bin, out of which M are black and $N - M$ are white. At each step, we draw a ball at random without replacement. Consider the random variable Y describing the total number of balls in our sample after we collect the x-th black ball. This random variable follows the so-called negative hypergeometric (NHG) distribution. Using our representation of an order-preserving function, it is not hard to show that $f(x)$ for a given point $x \in \{1, \ldots, M\}$ has a NHG distribution over a random choice of f. Assuming an efficient sampling algorithm for the NHG distribution, this gives a rough idea for a scheme, but there are still many subtleties to take care of.

Handling multiple points. First, assigning multiple plaintexts to ciphertexts independently according to the NHG distribution cannot work, because the resulting encryption function is unlikely to even be order-preserving. One could try to fix this by keeping tracking of all previously encrypted plaintexts and their ciphertexts (in both the encryption and decryption algorithms) and adjusting the parameters of the NHG sampling algorithm appropriately for each new plaintext. But we want a stateless scheme, so it cannot keep track of such previous assignments.

Eliminating the state. As a first step towards eliminating the state, we show that by assigning ciphertexts to plaintexts in a more organized fashion, the state can actually consist of a static but exponentially long random tape. The idea is that, to encrypt plaintext x, the encryption algorithm performs a binary search down to x. That is, it first assigns $\mathcal{E}nc(K, M/2)$, then $\mathcal{E}nc(K, M/4)$ if $m < M/2$ and $\mathcal{E}nc(K, 3M/4)$ otherwise, and so on, until $\mathcal{E}nc(K, x)$ is assigned. Crucially, each ciphertext assignment is made according to the output of the NHG sampling algorithm run on appropriate parameters and *coins from an associated portion of the random tape indexed by the plaintext*. (The decryption algorithm can be defined similarly.) Now, it may not be clear that the resulting scheme induces a *random* order-preserving function from the plaintext to ciphertext-space (does its distribution get skewed by the binary search?), but we prove (by strong induction on the size of the plaintext-space) that this is indeed the case.

Of course, instead of making the long random tape the secret key K for our scheme, we can make it the key for a PRF and generate portions of the tape dynamically as needed. However, coming up with a practical PRF construction to use here requires some care. For efficiency it should be blockcipher-based. Since the size of parameters to the NHG sampling algorithm as well as the number

of random coins it needs varies during the binary search, and also because such a construction seems useful in general, it should be both variable input-length (VIL) and variable output-length, which we call a *length-flexible* (LF)-PRF. We propose a generic construction of an LF-PRF from a VIL-PRF and a (keyless) VOL-PRG (pseudorandom generator). Efficient blockcipher-based VIL-PRFs are known, and we suggest a highly efficient blockcipher-based VOL-PRG that is apparently folklore. POPF-CCA security of the resulting OPE scheme can then be easily proved assuming only standard security (pseudorandomness) of an underlying blockcipher.

Switching from NHG to HG. Finally, our scheme needs an efficient sampling algorithm for the NHG distribution. Unfortunately, the existence of such an algorithm seems open. It is known that NHG can be approximated by the negative binomial distribution [26], which in turn can be sampled efficiently [16, 14], and that the approximation improves as M and N grow. However, quantifying the quality of approximation for fixed parameters seems difficult.

Instead, we turn to a related probability distribution, namely the hypergeometric (HG) distribution, for which a very efficient exact (not approximated) sampling algorithm is known [22, 23]. In our balls-and-bin model with M black and $N - M$ white balls, the random variable X specifying the number of black balls in our sample as soon as y balls are picked follows the HG distribution. The scheme based on this distribution, which is the one described in the body of the paper, is rather more involved, but nearly as efficient: instead of $O(\log M) \cdot T_{\mathrm{NHGD}}$ running-time it is $O(\log N) \cdot T_{\mathrm{HGD}}$ (where $T_{\mathrm{NHGD}}, T_{\mathrm{HGD}}$ are the running-times of the sampling algorithms for the respective distributions), but we show that it is $O(\log M) \cdot T_{\mathrm{HGD}}$ on average.

We note that the hypergeometric distribution was also used in [19] for sampling pseudorandom permutations and constructing blockciphers for short inputs. The authors of [19] were unaware of the efficient sampling algorithms for HG [22, 23] and provided their own realizations based on general sampling methods.

Discussion. It is important to realize that the "ideal" object in our POPF-CCA definition (a random order-preserving function), and correspondingly our OPE construction meeting it, inherently leak some information about the underlying plaintexts. Characterizing this leakage is an important next step in the study of OPE but is outside the scope of our current paper. (Although we mention that our "big-jump attack" of Theorem 3.1 may provide some insight in this regard.)

The point is that practitioners have indicated their desire to use OPE schemes in order to achieve efficient range queries on encrypted data and are willing to live with its security limitations. In response, we provide a scheme meeting what we believe to be a "best-possible" security notion for OPE. This belief can be justified by noting that it is usually the case that a security notion for a cryptographic object is met by a "random" one (which is sometimes built directly into the definition, as in the case of PRFs and PRPs).

But before one fully understands how the security properties of the ideal object, a random order-preserving function, fit the security needs of applications, we *do not recommend* the practical use of our construction.

On a more general primitive. To allow efficient range queries on encrypted data, it is sufficient to have an order-preserving hash function family H (not necessarily invertible). The overall OPE scheme would then have secret key $(K_{\mathcal{E}nc}, K_H)$ where $K_{\mathcal{E}nc}$ is a key for a normal (randomized) encryption scheme and K_H is a key for H, and the encryption of x would be $\mathcal{E}nc(K_{\mathcal{E}nc}, x) \| H(K_H, x)$ (cf. efficiently searchable encryption (ESE) in [6]). Our security notion (in the CPA case) can also be applied to such H. In fact, there has been some work on hash functions that are order-preserving or have some related properties [25, 15, 20]. But none of these works are concerned with security in any sense. Since our

OPE scheme is efficient and already invertible, we have not tried to build any secure order-preserving hash separately.

On the public-key setting. Finally, it is interesting to note that in a public-key setting one cannot expect OPE to provide any privacy at all. Indeed, given a ciphertext c computed under public key pk, anyone can decrypt c via a simple binary-search. In the symmetric-key setting a real-life adversary cannot simply encrypt messages itself, so such an attack is unlikely to be feasible.

2 Preliminaries

Notation and conventions. We refer to members of $\{0,1\}^*$ as strings. If x is a string then $|x|$ denotes its length in bits and if x, y are strings then $x\|y$ denotes an encoding from which x, y are uniquely recoverable. For $\ell \in \mathbb{N}$ we denote by 1^ℓ the string of ℓ "1" bits. If S is a set then $x \xleftarrow{\$} S$ denotes that x is selected uniformly at random from S. For convenience, for any $k \in \mathbb{N}$ we write $x_1, x_2, \ldots, x_k \xleftarrow{\$} S$ as shorthand for $x_1 \xleftarrow{\$} S, x_2 \xleftarrow{\$} S, \ldots, x_n \xleftarrow{\$} S$. If A is a randomized algorithm and Coins is the set from where it draws its coins, then we write $A(x, y, \ldots)$ as shorthand for $R \xleftarrow{\$} \mathsf{Coins}\,;\ A(x, y, \ldots; R)$, where the latter denotes the result of running A on inputs $x, y, \ldots$ and coins R. And $a \xleftarrow{\$} A(x, y, \ldots)$ means that we assign to a the output of A run on inputs $x, y, \ldots$. For $a \in \mathbb{N}$ we denote by $[a]$ the set $\{1, \ldots, a\}$. For sets X and Y, if $f\colon X \to Y$ is a function, then we call X the domain, Y the range, and the set $\{f(x) \mid x \in X\}$ the image of the function. An adversary is an algorithm. By convention, all algorithms are required to be efficient, meaning run in (expected) polynomial-time in the length of their inputs, and their running-time includes that of any overlying experiment.

Symmetric Encryption. A *symmetric encryption scheme* $\mathcal{SE} = (\mathcal{K}, \mathcal{E}nc, \mathcal{D}ec)$ with associated *plaintext-space* $\mathcal{D}$ and *ciphertext-space* $\mathcal{R}$ consists of three algorithms.

- The randomized *key generation algorithm* $\mathcal{K}$ returns a secret key K.
- The (possibly randomized) *encryption algorithm* $\mathcal{E}nc$ takes the secret key K, descriptions of plaintext and ciphertext-spaces $\mathcal{D}, \mathcal{R}$ and a plaintext m to return a ciphertext c.
- The deterministic *decryption algorithm* $\mathcal{D}ec$ takes the secret key K, descriptions of plaintext and ciphertext-spaces $\mathcal{D}, \mathcal{R}$, and a ciphertext c to return a corresponding plaintext m or a special symbol $\perp$ indicating that the ciphertext was invalid.

Note that the above syntax differs from the usual one in that we specify the plaintext and ciphertext-spaces $\mathcal{D}, \mathcal{R}$ explicitly; this is for convenience relative to our specific schemes. We require the usual correctness condition, namely that $\mathcal{D}ec(K, \mathcal{D}, \mathcal{R}, (\mathcal{E}nc(K, \mathcal{D}, \mathcal{R}, m)) = m$ for all K output by $\mathcal{K}$ and all $m \in \mathcal{D}$. Finally, we say that $\mathcal{SE}$ is *deterministic* if $\mathcal{E}nc$ is deterministic.

IND-CPA. Let $\mathcal{LR}(\cdot, \cdot, b)$ denote the function that on inputs m_0, m_1 returns m_b. For a symmetric encryption scheme $\mathcal{SE} = (\mathcal{K}, \mathcal{E}nc, \mathcal{D}ec)$ and an adversary A and $b \in \{0,1\}$ consider the following experiment:

Experiment $\mathbf{Exp}^{\text{ind-cpa-b}}_{\mathcal{SE}}(A)$
$K \xleftarrow{\$} \mathcal{K}$
$d \xleftarrow{\$} A^{\mathcal{E}nc(K, \mathcal{LR}(\cdot,\cdot,b))}$
Return d

We require that each query (m_0, m_1) that A makes to its oracle satisfies $|m_0| = |m_1|$. For an adversary A, define its *ind-cpa advantage* against $\mathcal{SE}$ as

$$\mathbf{Adv}^{\text{ind-cpa}}_{\mathcal{SE}}(A) \;=\; \Pr[\,\mathbf{Exp}^{\text{ind-cpa-1}}_{\mathcal{SE}}(A) = 1\,] - \Pr[\,\mathbf{Exp}^{\text{ind-cpa-0}}_{\mathcal{SE}}(A) = 1\,]\;.$$

Pseudorandom functions (PRFs). A *family of functions* is a map $F\colon \mathit{Keys} \times \mathcal{D} \to \{0,1\}^\ell$, where for each key $K \in \mathit{Keys}$ the map $F(K,\cdot)\colon \mathcal{D} \to \{0,1\}^\ell$ is a function. We refer to $F(K,\cdot)$ as an *instance* of F. For an adversary A, its *prf-advantage* against F, $\mathbf{Adv}^{\text{prf}}_F(A)$, is defined as

$$\Pr\left[\; K \xleftarrow{\$} \mathit{Keys} \;:\; A^{F(K,\cdot)} = 1 \;\right] - \Pr\left[\; f \xleftarrow{\$} \mathsf{Func}_{\mathcal{D},\{0,1\}^\ell} \;:\; A^{f(\cdot)} = 1 \;\right]\;,$$

where $\mathsf{Func}_{\mathcal{D},\{0,1\}^\ell}$ denotes the set of all functions from $\mathcal{D}$ to $\{0,1\}^\ell$.

3 OPE and its Security

3.1 Order-Preserving Encryption (OPE)

We are interested in deterministic encryption schemes that preserve numerical ordering on their plaintext-space. Let us define what we mean by this. For $A, B \subseteq \mathbb{N}$ with $|A| \leq |B|$, a function $f\colon A \to B$ is *order-preserving* (aka. strictly-increasing) if for all $i, j \in A$, $f(i) > f(j)$ iff $i > j$. We say that deterministic encryption scheme $\mathcal{SE} = (\mathcal{K}, \mathcal{E}nc, \mathcal{D}ec)$ with plaintext and ciphertext-spaces $\mathcal{D}, \mathcal{R}$ is *order-preserving* if $\mathcal{E}nc(K,\cdot)$ is an order-preserving function from $\mathcal{D}$ to $\mathcal{R}$ for all K output by $\mathcal{K}$ (with elements of $\mathcal{D}, \mathcal{R}$ interpreted as numbers, encoded as strings). Unless otherwise stated, we assume the plaintext-space is $[M]$ and the ciphertext-space is $[N]$ for some $N \geq M \in \mathbb{N}$.

3.2 Security of OPE

A first try. Security of deterministic symmetric encryption was introduced in [8], as a notion they call *security under distinct chosen-plaintext attack (IND-DCPA)*. (It will not be important to consider CCA now.) The idea is that because deterministic encryption leaks plaintext equality, the adversary A in the IND-CPA experiment defined in Section 2 is restricted to make only *distinct* queries on either side of its oracle (as otherwise there is a trivial attack). That is, supposing A makes queries $(m_0^1, m_1^1), \ldots, (m_0^q, m_1^q)$, they require that $m_b^1, \ldots m_b^q$ are all distinct for $b \in \{0,1\}$.

Noting that any OPE scheme analogously leaks the order relations among the plaintexts, let us first try generalizing the above approach to take this into account. Namely, let us further require the above queries made by A to satisfy $m_0^i < m_0^j$ iff $m_1^i < m_1^j$ for all $1 \leq i, j \leq q$. We call such an A an *IND-OCPA adversary* for *indistinguishability under ordered chosen-plaintext attack*.

IND-OCPA is not useful. Defining IND-OCPA adversary seems like a plausible way to analyze security for OPE. Surprisingly, it turns out not to be too useful for us. Below, we show that IND-OCPA is unachievable by a practical order-preserving encryption scheme, in that an OPE scheme cannot be IND-OCPA unless its ciphertext-space is extremely large (exponential in the size of the plaintext-space).

Theorem 3.1 *Let $\mathcal{SE} = (\mathcal{K}, \mathcal{E}nc, \mathcal{D}ec)$ be an order-preserving encryption scheme with plaintext-space $[M]$ and ciphertext-space $[N]$ for $M, N \in \mathbb{N}$ such that $2^{k-1} \leq N < 2^k$ for some $k \in \mathbb{N}$. Then there*

exists an IND-OCPA adversary A against $\mathcal{SE}$ such that

$$\mathbf{Adv}^{\text{ind-cpa}}_{\mathcal{SE}}(A) \;\geq\; 1 - \frac{2k}{M-1}\;.$$

Furthermore, A runs in time $O(\log N)$ and makes 3 oracle queries. ▌

So, k in the theorem should be almost as large as M for A's advantage to be small.

We introduce the following concept for the proof. For an order-preserving function $f\colon [M] \to [N]$ call $i \in \{3,\ldots,M-1\}$ a *big jump* of f if the f-distance to the next point is as big as the sum of all the previous, i.e. $f(i+1) - f(i) \geq f(i) - f(1)$. Similarly we call $i \in \{2,\ldots M-2\}$ a *big reverse-jump* of f if $f(i) - f(i-1) \geq f(M) - f(i)$. The proof uses the following simple combinatorial lemma.

Lemma 3.2 *Let $f\colon [M] \to [N]$ be an order-preserving function and suppose that f has k big jumps (respectively big reverse-jumps). Then $N \geq 2^k$.*

Proof: (of Lemma 3.2) Let $J = \{j_1,\ldots,j_k\}$ be the set of big jumps of f. (Our proof trivially adjusts to the case of big reverse jumps, so we do not address it separately.) We prove by induction that for every $1 \leq i \leq M-1$

$$f(j_i) \geq 2^i + f(1)\;.$$

Since $f(N) \geq f(j_k)$, the statement of the lemma follows.

The base case ($i = 1$) holds since $f(j_1) \geq 2 + f(1)$ is true by the definition of a big jump.

Assume $f(j_i) \geq 2^i + f(1)$ is true for $i = n$. We show that it is also true for $i = n+1$. We claim that

$$\begin{aligned} f(j_{n+1}) &\geq 2f(j_{n+1}-1) - f(1) \\ &\geq 2f(j_n) - f(1) \\ &\geq 2\cdot(2^n + f(1)) - f(1) \\ &= 2^{n+1} + f(1)\;. \end{aligned}$$

Above, the first inequality is by definition. The second uses that $j_{n+1} - 1 \geq j_n$, which is true because f is order-preserving and the range is $[N]$. The third is by the induction hypothesis. ▌

We now move on to the proof of the theorem.

Proof: (of Theorem 3.1) Consider the following ind-ocpa adversary A against $\mathcal{SE}$:

Adversary $A^{\mathcal{E}nc(K,\mathcal{LR}(\cdot,\cdot,b))}$
- $m \xleftarrow{\$} \{1,\ldots,M-1\}$
- $c_1 \leftarrow \mathcal{E}nc(K, \mathcal{LR}(1, m, b))$
- $c_2 \leftarrow \mathcal{E}nc(K, \mathcal{LR}(m, m+1, b))$
- $c_2 \leftarrow \mathcal{E}nc(K, \mathcal{LR}(m+1, M, b))$
- Return 1 if $(c_3 - c_2) > (c_2 - c_1)$
- Else return 0

First we claim that

$$\Pr[\,\mathbf{Exp}^{\text{ind-ocpa-1}}_{\mathcal{SE}}(A) = 1\,] \;\geq\; \frac{(M-1)-k}{M-1} \;=\; 1 - \frac{k}{M-1}\;.$$

The reason is that m is picked independently at random and if $b = 1$ then A outputs 1 just when $m+1$ is not a big reverse-jump of $\mathcal{E}nc(K,\cdot)$, and since $N \leq 2^k$ we know that $\mathcal{E}nc(K,\cdot)$ has at most k big reverse-jumps by Lemma 3.2. Similarly,

$$\Pr[\,\mathbf{Exp}^{\text{ind-ocpa-0}}_{\mathcal{SE}}(A) = 1\,] \;\leq\; \frac{k}{M-1}$$

because if $b = 0$ then A outputs 1 just when m is a big jump of $\mathcal{E}nc(K,\cdot)$, and since $N \leq 2^k$ we know that $\mathcal{E}nc(K,\cdot)$ has at most k big jumps by Lemma 3.2. Subtracting yields the theorem. Note that A only needs to pick a random element of $[M]$ and do basic operations on elements of $[N]$, which is $O(\log N)$ as claimed. ∎

Discussion. The adversary in the proof of Theorem 3.1 uses what we call the "big-jump attack" to distinguish between ciphertexts of messages that are "very close" and "far apart." The attack shows that *any* practical OPE scheme inherently leaks more information about the plaintexts than just their ordering, namely some information about their relative distances. We return to this point later.

An alternative approach. Instead, we take the approach used in defining security e.g. of PRPs [17] or on-line PRPs [5], where one asks that oracle access to the function in question be indistinguishable from access to the corresponding "ideal" random object, e.g. a random permutation or a random on-line permutation. As order-preserving functions are injective, we consider the "strong" version of such a definition where an inverse oracle is also given.

POPF-CCA. Fix an order-preserving encryption scheme $\mathcal{SE} = (\mathcal{K}, \mathcal{E}nc, \mathcal{D}ec)$ with plaintext-space $\mathcal{D}$ and ciphertext-space $\mathcal{R}$, $|\mathcal{D}| \leq |\mathcal{R}|$. For an adversary A against $\mathcal{SE}$, define its *popf-cca-advantage* (or *pseudorandom order-preserving function advantage under chosen-ciphertext attack*), $\mathbf{Adv}^{\text{popf-cca}}_{\mathcal{SE}}(A)$, against $\mathcal{SE}$ as

$$\Pr\left[\, K \xleftarrow{\$} \mathcal{K} \;:\; A^{\mathcal{E}nc(K,\cdot),\mathcal{D}ec(K,\cdot)} = 1 \,\right] - \Pr\left[\, g \xleftarrow{\$} \mathsf{OPF}_{\mathcal{D},\mathcal{R}} \;:\; A^{g(\cdot),g^{-1}(\cdot)} = 1 \,\right] ,$$

where $\mathsf{OPF}_{\mathcal{D},\mathcal{R}}$ denotes the set of all order-preserving functions from $\mathcal{D}$ to $\mathcal{R}$.

Lazy sampling. Now in order for this notion to be useful, i.e. to be able show that a scheme achieves it, we also need a way to implement A's oracles in the "ideal" experiment efficiently. In other words, we need to show how to "lazy sample" (a term from [9]) a random order-preserving function and its inverse.[1]

As shown in [9], lazy sampling of "exotic" functions with many constraints can be tricky. In the case of a random order-preserving function, it turns out that straightforward procedures—which assign a random point in the range to a queried domain point, subject to the obvious remaining constraints—do not work (that is, the resulting function is not uniformly distributed over the set of all such functions). So how can we lazy sample such a function, if it is possible at all? We address this issue next.

A caveat. Before proceeding, we note that a shortcoming of our POPF-CCA notion is it does not lead to a nice answer to the question of what information about the data is leaked by a secure OPE scheme, but only reduces this to the question of what information the "ideal object" (a random order-preserving function) leaks. Although practitioners have indicated that they are willing to live with

[1] For example, in the case of a random function from the set of *all* functions one can simply assign a random point from the range to each new point queried from the domain. In the case of a random permutation, the former can be chosen from the set of all previously unassigned points in the range, and lazy sampling of its inverse can be done similarly. A lazy sampling procedure for a random on-line PRP and its inverse via a tree-based characterization was given in [5].

the security limitations of OPE for its useful functionality, more precisely characterizing the latter remains an important next step before our schemes should be considered for practical deployment.

4 Lazy Sampling a Random Order-Preserving Function

In this section, we show how to lazy-sample a random order-preserving function and its inverse. This result may also be of independent interest, since the more general question of what functions can be lazy-sampled is interesting in its own right, and it may find other applications as well, e.g. to [12]. We first uncover a connection between a random order-preserving function and the hypergeometric (HG) probability distribution.

4.1 The Hypergeometric Connection

To gain some intuition we start with the following claim.

Proposition 4.1 *There is bijection between the set* $\mathsf{OPF}_{\mathcal{D},\mathcal{R}}$ *containing all order-preserving functions from a domain* $\mathcal{D}$ *of size* M *to a range* $\mathcal{R}$ *of size* $N \geq M$ *and the set of all possible combinations of* M *out of* N *ordered items.*

Proof: Without loss of generality, it is enough to prove the result for domain $[M]$ and range $[N]$. Imagine a graph with its x-axis marked with integers from 1 to M and its $y = f(x)$-axis marked with integers from 1 to N. Given S, a set of M distinct integers from $[N]$, construct an order-preserving function from $[M]$ to $[N]$ by mapping each $i \in [M]$ to the ith smallest element in S. So, an M-out-of-N combination corresponds to a unique order-preserving function. On the other hand, consider an order-preserving function f from $[M]$ to $[N]$. The image of f defines a set of M distinct objects in $[N]$, so an order-preserving function corresponds to a unique M-out-of-N combination. ∎

Using the above combination-based characterization it is straightforward to justify the following equality, defined for $M, N \in \mathbb{N}$ and any $x, x+1 \in [M], y \in [N]$:

$$\Pr[\, f(x) \leq y < f(x+1) \colon f \xleftarrow{\$} \mathsf{OPF}_{[M],[N]} \,] = \frac{\binom{y}{x} \cdot \binom{N-y}{M-x}}{\binom{N}{M}} \,. \tag{1}$$

Now let us recall a particular distribution dealing with an experiment of selecting from combinations of items.

Hypergeometric distribution. Consider the following balls-and-bins model. Assume we have N balls in a bin out of which M balls are black and $N - M$ balls are white. At each step we draw a ball at random, without replacement. Consider a random variable X that describes the number of black balls chosen after a *sample size* of y balls are picked. This random variable has a hypergeometric distribution, and the probability that $X = x$ for the parameters N, M, y is

$$P_{HGD}(x; N, M, y) = \frac{\binom{y}{x} \cdot \binom{N-y}{M-x}}{\binom{N}{M}} \,.$$

Notice the equality to the right hand side of Equation (1). Intuitively, this equality means we can view constructing a random order–preserving function f from $[M]$ to $[N]$ as an experiment where we have N balls, M of which are black. Choosing balls randomly without replacement, if the y-th ball

we pick is black then the least unmapped point in the domain is mapped to y under f. Of course, this experiment is too inefficient to be performed directly. But we will use the hypergeometric distribution to design procedures that efficiently and recursively lazy sample a random order-preserving function and its inverse.

4.2 The LazySample Algorithms

Here we give our algorithms **LazySample**, **LazySampleInv** that lazy sample a random order-preserving function from domain $\mathcal{D}$ to range $\mathcal{R}$, $|\mathcal{D}| \leq |\mathcal{R}|$, and its inverse, respectively. The algorithms share and maintain joint state. We assume that both $\mathcal{D}$ and $\mathcal{R}$ are sets of consecutive integers.

TWO SUBROUTINES. Our algorithms make use of two subroutines. The first, denoted HGD, takes inputs $\mathcal{D}, \mathcal{R}$, and $y \in \mathcal{R}$ to return $x \in \mathcal{D}$ such that for each $x^* \in \mathcal{D}$ we have $x = x^*$ with probability $P_{HGD}(x-d; |\mathcal{R}|, |\mathcal{D}|, y-r)$ over the coins of HGD, where $d = \min(\mathcal{D})-1$ and $r = \min(\mathcal{R})-1$. (Efficient algorithms for this exist, and we discuss them in Section 4.5.) The second, denoted GetCoins, takes inputs 1^ℓ,$\mathcal{D}$,$\mathcal{R}$, and $b\|z$, where $b \in \{0,1\}$ and $z \in \mathcal{R}$ if $b = 0$ and $z \in \mathcal{D}$ otherwise, to return $cc \in \{0,1\}^\ell$.

THE ALGORITHMS. To define our algorithms, let us denote by $w \xleftarrow{cc} S$ that w is assigned a value sampled uniformly at random from set S using coins cc of length ℓ_S, where ℓ_S denotes the number of coins needed to do so. Let $\ell_1 = \ell(\mathcal{D}, \mathcal{R}, y)$ denote the number of coins needed by HGD on inputs $\mathcal{D}, \mathcal{R}, y$. Our algorithms are given in Figure 1. Note that the arrays F, I, initially empty, are global and shared between the algorithms; also, for now, think of GetCoins as returning fresh random coins. We later implement it by using a PRF on the same parameters to eliminate the joint state.

OVERVIEW. To determine the image of input m, **LazySample** employs a strategy of mapping "range gaps" to "domain gaps" in a recursive, binary search manner. By "range gap" or "domain gap," we mean an imaginary barrier between two consecutive points in the range or domain, respectively. When run, the algorithm first maps the middle range gap y (the gap between the middle two range points) to a domain gap. To determine the mapping, on line 11 it sets, according to the hypergeometric distribution, how many points in $\mathcal{D}$ are mapped up to range point y and stores this value in array I. (In the future the array is referenced instead of choosing this value anew.) Thus we have that $f(x) \leq y < f(x+1)$ (cf. Equation (1)), where $x = d + I[\mathcal{D}, \mathcal{R}, y]$ as computed on line 12. So, we can view the range gap between y and $y+1$ as having been mapped to the domain gap between x and $x+1$.

If the input domain point m is below (resp. above) the domain gap, the algorithm recurses on line 19 on the lower (resp. upper) half of the range and the lower (resp. upper) part of the domain, mapping further "middle" range gaps to domain gaps. This process continues until the gaps on either side of m have been mapped to by some range gaps. Finally, on line 07, the algorithm samples a range point uniformly at random from the "window" defined by the range gaps corresponding to m's neighboring domain gaps. The is result assigned to array F as the image of m under the lazy-sampled function.

4.3 Correctness

When GetCoins returns truly random coins, it is not hard to observe that **LazySample**, **LazySampleInv** are consistent and sample an order-preserving function and its inverse respectively. But we need a stronger claim; namely, that our algorithms sample a *random* order-preserving function and its inverse. We show this by arguing that any (even computationally unbounded) adversary has no

LazySample$(\mathcal{D}, \mathcal{R}, m)$

$M \leftarrow |\mathcal{D}|$; $N \leftarrow |\mathcal{R}|$
$d \leftarrow \min(\mathcal{D}) - 1$; $r \leftarrow \min(\mathcal{R}) - 1$
$y \leftarrow r + \lceil N/2 \rceil$
If $|\mathcal{D}| = 1$ then
    If $F[\mathcal{D}, \mathcal{R}, m]$ is undefined then
        $cc \xleftarrow{\$} \text{GetCoins}(1^{\ell_{\mathcal{R}}}, \mathcal{D}, \mathcal{R}, 1 \| m)$
        $F[\mathcal{D}, \mathcal{R}, m] \xleftarrow{cc} \mathcal{R}$
    Return $F[\mathcal{D}, \mathcal{R}, m]$

If $I[\mathcal{D}, \mathcal{R}, y]$ is undefined then
    $cc \xleftarrow{\$} \text{GetCoins}(1^{\ell_1}, \mathcal{D}, \mathcal{R}, 0 \| y)$
    $I[\mathcal{D}, \mathcal{R}, y] \xleftarrow{\$} \text{HGD}(\mathcal{D}, \mathcal{R}, y; cc)$
$x \leftarrow I[\mathcal{D}, \mathcal{R}, y]$
If $m \leq x$ then
    $\mathcal{D} \leftarrow \{d+1, \ldots, x\}$
    $\mathcal{R} \leftarrow \{r+1, \ldots, y\}$
Else
    $\mathcal{D} \leftarrow \{x+1, \ldots, d+M\}$
    $\mathcal{R} \leftarrow \{y+1, \ldots, r+N\}$
Return **LazySample**$(\mathcal{D}, \mathcal{R}, m)$

LazySampleInv$(\mathcal{D}, \mathcal{R}, c)$

$M \leftarrow |\mathcal{D}|$; $N \leftarrow |\mathcal{R}|$
$d \leftarrow \min(\mathcal{D}) - 1$; $r \leftarrow \min(\mathcal{R}) - 1$
$y \leftarrow r + \lceil N/2 \rceil$
If $|\mathcal{D}| = 1$ then $m \leftarrow \min(\mathcal{D})$
    If $F[\mathcal{D}, \mathcal{R}, m]$ is undefined then
        $cc \xleftarrow{\$} \text{GetCoins}(1^{\ell_{\mathcal{R}}}, \mathcal{D}, \mathcal{R}, 1 \| m)$
        $F[\mathcal{D}, \mathcal{R}, m] \xleftarrow{cc} \mathcal{R}$
    If $F[\mathcal{D}, \mathcal{R}, m] = c$ then return m
    Else return $\perp$
If $I[\mathcal{D}, \mathcal{R}, y]$ is undefined then
    $cc \xleftarrow{\$} \text{GetCoins}(1^{\ell_1}, \mathcal{D}, \mathcal{R}, 0 \| y)$
    $I[\mathcal{D}, \mathcal{R}, y] \xleftarrow{\$} \text{HGD}(\mathcal{D}, \mathcal{R}, y; cc)$
$x \leftarrow I[\mathcal{D}, \mathcal{R}, y]$
If $c \leq y$ then
    $\mathcal{D} \leftarrow \{d+1, \ldots, x\}$
    $\mathcal{R} \leftarrow \{r+1, \ldots, y\}$
Else
    $\mathcal{D} \leftarrow \{x+1, \ldots, d+M\}$
    $\mathcal{R} \leftarrow \{y+1, \ldots, r+N\}$
Return **LazySampleInv**$(\mathcal{D}, \mathcal{R}, c)$

Figure 1: The **LazySample, LazySampleInv** algorithms.

advantage in distinguishing oracle access to a random order-preserving function and its inverse from that to the algorithms **LazySample**, **LazySampleInv**. The following theorem states this claim.

Theorem 4.2 *Suppose GetCoins returns truly random coins on each new input. Then for any (even computationally unbounded) algorithm A we have*

$$\Pr[\, A^{g(\cdot),g^{-1}(\cdot)} = 1 \,] \;=\; \Pr[\, A^{\textbf{\textit{LazySample}}(\mathcal{D},\mathcal{R},\cdot),\textbf{\textit{LazySampleInv}}(\mathcal{D},\mathcal{R},\cdot)} = 1 \,] \,,$$

where g, g^{-1} denote an order-preserving function picked at random from $\mathsf{OPF}_{\mathcal{D},\mathcal{R}}$ and its inverse, respectively. ∎

Proof: Since we consider unbounded adversaries, we can ignore the inverse oracle in our analysis, since such an adversary can always query all points in the domain to learn all points in the image. Let $M = |\mathcal{D}|$, $N = |\mathcal{R}|$, $d = \min(\mathcal{D}) - 1$, and $r = \min(\mathcal{R}) - 1$. We will say that two functions $g, h : \mathcal{D} \to \mathcal{R}$ are *equivalent* if $g(m) = h(m)$ for all $m \in \mathcal{D}$. (Note that if $\mathcal{D} = \emptyset$, any two functions $g, h : \mathcal{D} \to \mathcal{R}$ are vacuously equivalent.) Let f be any function in $\mathsf{OPF}_{\mathcal{D},\mathcal{R}}$. To prove the theorem, it is enough to show that the function defined by $\textbf{LazySample}(\mathcal{D}, \mathcal{R}, \cdot)$ is equivalent to f with probability $1/|\mathsf{OPF}_{\mathcal{D},\mathcal{R}}|$. We prove this using strong induction on M and N.

Consider the base case where $M = 1$, i.e., $\mathcal{D} = \{m\}$ for some m, and $N \geq M$. When it is first called, $\textbf{LazySample}(\mathcal{D}, \mathcal{R}, m)$ will determine an element c uniformly at random from $\mathcal{R}$ and enter it into $F[\mathcal{D}, \mathcal{R}, m]$, whereupon any future calls of $\textbf{LazySample}(\mathcal{D}, \mathcal{R}, m)$ will always output $F[\mathcal{D}, \mathcal{R}, m] = c$. Thus, the output of $\textbf{LazySample}(\mathcal{D}, \mathcal{R}, m)$ is always c, so $\textbf{LazySample}(\mathcal{D}, \mathcal{R}, \cdot)$ is equivalent to f if and only if $c = f(m)$. Since c is chosen randomly from $\mathcal{R}$, $c = f(m)$ with probability $1/|\mathcal{R}|$. Thus, $\textbf{LazySample}(\mathcal{D}, \mathcal{R}, m)$ is equivalent to $f(m)$ with probability $1/|\mathcal{R}| = 1/|\mathsf{OPF}_{\mathcal{D},\mathcal{R}}|$.

Now suppose $M > 1$, and $N \geq M$. As an induction hypothesis assume that for all domains $\mathcal{D}'$ of size M' and ranges $\mathcal{R}'$ of size $N' \geq M'$, where either $M' < M$ or ($M' = M$ and $N' < N$), and for any function f' in $\mathsf{OPF}_{\mathcal{D}',\mathcal{R}'}$, $\textbf{LazySample}(\mathcal{D}', \mathcal{R}', \cdot)$ is equivalent to f' with probability $1/|\mathsf{OPF}_{\mathcal{D}',\mathcal{R}'}|$.

The first time it is called, $\textbf{LazySample}(\mathcal{D}, \mathcal{R}, \cdot)$ first computes $I[\mathcal{D}, \mathcal{R}, y] \xleftarrow{\$} \mathrm{HGD}(\mathcal{R}, \mathcal{D}, y - r)$, where $y = r + \lceil N/2 \rceil$, $r = \min(\mathcal{R}) - 1$. Henceforth, on this and future calls of $\textbf{LazySample}(\mathcal{D}, \mathcal{R}, m)$, the algorithm sets $x = d + I[\mathcal{D}, \mathcal{R}, y - r]$ and will run $\textbf{LazySample}(\mathcal{D}_1, \mathcal{R}_1, m)$ if $m \leq x$, or run $\textbf{LazySample}(\mathcal{D}_2, \mathcal{R}_2, m)$ if $m > x$, where $\mathcal{D}_1 = \{1, \ldots, x\}$, $\mathcal{R}_1 = \{1, \ldots, y\}$, $\mathcal{D}_2 = \{x + 1, \ldots, M\}$, $\mathcal{R}_2 = \{y+1, \ldots, N\}$. Let f_1 be f restricted to the domain $\mathcal{D}_1$, and let f_2 be f restricted to the domain $\mathcal{D}_2$. Let x_0 be the unique integer in $\mathcal{D} \cup \{d\}$ such that $f(z) \leq y$ for all $z \in \mathcal{D}$, $z \leq x_0$, and $f(z) > y$ for all $z \in \mathcal{D}$, $z > x_0$. Note then that $\textbf{LazySample}(\mathcal{D}, \mathcal{R}, \cdot)$ is equivalent to f if and only if all three of the following events occur:

- E_1: f restricted to range $\mathcal{R}_1$ stays within domain $\mathcal{D}_1$, and f restricted to range $\mathcal{R}_2$ stays within domain $\mathcal{D}_2$—that is, x is chosen to be x_0.
- E_2: $\textbf{LazySample}(\mathcal{D}_1, \mathcal{R}_1, \cdot)$ is equivalent to f_1.
- E_3: $\textbf{LazySample}(\mathcal{D}_2, \mathcal{R}_2, \cdot)$ is equivalent to f_2.

By the law of conditional probability, and since E_2 and E_3 are independent,

$$\Pr[\, E_1 \cap E_2 \cap E_3 \,] = \Pr[\, E_1 \,] \cdot \Pr[\, E_2 \cap E_3 \mid E_1 \,] = \Pr[\, E_1 \,] \cdot \Pr[\, E_2 \mid E_1 \,] \cdot \Pr[\, E_3 \mid E_1 \,] \;.$$

$\Pr[\, E_1 \,]$ is the hypergeometric probability that $\mathrm{HGD}(\mathcal{R}, \mathcal{D}, y - r)$ will return $x_0 - d$, so

$$\Pr[\, E_1 \,] = P_{HGD}(x_0 - d; N, M, \lceil N/2 \rceil) = \frac{\binom{\lceil N/2 \rceil}{x_0 - d}\binom{N - \lceil N/2 \rceil}{M - (x_0 - d)}}{\binom{N}{M}}.$$

Assuming for the moment that neither $\mathcal{D}_1$ nor $\mathcal{D}_2$ are empty, notice that both $|\mathcal{R}_1|$ and $|\mathcal{R}_2|$ are strictly less than $|\mathcal{R}|$, and $|\mathcal{D}_1|$ and $|\mathcal{D}_2|$ are less than or equal to $|\mathcal{D}|$, so the induction hypothesis holds for each. That is, $\textbf{LazySample}(\mathcal{D}_1, \mathcal{R}_1, \cdot)$ is equivalent to f_1 with probability $1/|\mathsf{OPF}_{\mathcal{D}_1,\mathcal{R}_1}| = 1/\binom{|\mathcal{R}_1|}{|\mathcal{D}_1|}$, and $\textbf{LazySample}(\mathcal{D}_2, \mathcal{R}_2, \cdot)$ is equivalent to f_2 with probability $1/|\mathsf{OPF}_{\mathcal{D}_2,\mathcal{R}_2}| = 1/\binom{|\mathcal{R}_2|}{|\mathcal{D}_2|}$. Thus, we have that

$$\Pr[\, E_2 \mid E_1 \,] = \frac{1}{\binom{\lceil N/2 \rceil}{x_0 - d}} \qquad \text{and} \qquad \Pr[\, E_3 \mid E_1 \,] = \frac{1}{\binom{N - \lceil N/2 \rceil}{d + M - x_0}}.$$

Also, note that if $\mathcal{D}_1 = \emptyset$, then $\Pr[\, E_2 \mid E_1 \,] = 1 = 1/\binom{\lceil N/2 \rceil}{x_0 - d}$ since $x_0 = d$. Likewise, if $\mathcal{D}_2 = \emptyset$, then $\Pr[\, E_3 \mid E_1 \,]$ will be the same as above. We conclude that

$$\Pr[\, E_1 \cap E_2 \cap E_3 \,] = \frac{\binom{\lceil N/2 \rceil}{x_0 - d}\binom{N - \lceil N/2 \rceil}{M - (x_0 - d)}}{\binom{N}{M}} \cdot \frac{1}{\binom{\lceil N/2 \rceil}{x_0 - d}} \cdot \frac{1}{\binom{N - \lceil N/2 \rceil}{d + M - x_0}} = \frac{1}{\binom{N}{M}}.$$

Therefore, $\textbf{LazySample}(\mathcal{D}, \mathcal{R}, \cdot)$ is equivalent to f with probability $1/\binom{N}{M} = 1/|\mathsf{OPF}_{\mathcal{D},\mathcal{R}}|$. Since f was an arbitrary element of $\mathsf{OPF}_{\mathcal{D},\mathcal{R}}$, the result follows. ▌

We clarify that in the theorem A's oracles for **LazySample**, **LazySampleInv** in the right-hand-side experiment share and update joint state. It is straightforward to check, via simple probability calculations, that the theorem holds for an adversary A that makes one query. The case of multiple queries is harder. The reason is that the distribution of the responses given to subsequent queries depends on which queries A has already made, and this distribution is difficult to compute directly. Instead our proof uses strong induction in a way that parallels the recursive nature of our algorithms.

4.4 Efficiency

We characterize efficiency of our algorithms in terms of the number of recursive calls made by **LazySample** or **LazySampleInv** before termination. (The proposition below is just stated in terms of **LazySample** for simplicity; the analogous result holds for **LazySampleInv**.)

Proposition 4.3 *The number of recursive calls made by* **LazySample** *is at most* $\log N + 1$ *in the worst-case and at most* $5 \log M + 12$ *on average.* ▌

Proof: For the worst case bound, note that **LazySample** performs a binary search over the range to map in the input domain point, on each recursion cutting the size of the possible range in half. Note that, by the nature of the hypergeometric probabilities, the size of the domain in each iteration can never exceed the size of the range. Thus, when the algorithm is called on a range of size 1, its domain is also of size 1, and the algorithm must terminate. Over the course of $\log N$ binary-search recursions, the range will shrink to size 1, so we conclude that a worst-case $\log N + 1$ recursions are required for **LazySample** to terminate.

For the average case bound, we use a result of Chvátal [**?**] that the tail of the hypergeometric distribution can be bounded so that

$$\sum_{i=k+1}^{M} P_{HGD}(i; N, M, c) \;\leq\; e^{-2t^2 M},$$

where t is a fraction such that $0 \leq t \leq 1 - c/N$, and $k = (c/N + t)M$. Taking $c = N/2$, this implies an upper bound on the probability of the hypergeometric distribution assigning our middle domain gap to an "outlying" domain gap:

$$\sum_{i \notin S} P_{HGD}(i; N, M, N/2) \;\leq\; 2e^{-2t^2 M} \tag{2}$$

where S is the subdomain $[(1/2 - t)M, (1/2 + t)M]$.

For $M < 12$, after at most 12 calls to **LazySample** we will reach a domain of size 1, and terminate. So suppose that $M \geq 12$. Taking $t = 1/4$ in Equation (2) implies that **LazySample** assigns the middle ciphertext gap to a plaintext gap in the "middle subdomain" $[M/4, 3M/4]$ with probability at least $1 - 2e^{-2(1/4)^2 M} \geq 1 - 2e^{-3/2} > 1/2$. When a domain gap in S is chosen it shrinks the current domain by a fraction of at least $3/4$. So, picking in the middle subdomain $\log_{4/3} M = \frac{\log M}{\log 4/3} < 2.5 \log M$ times will shrink it to size less than 12. Since the probability to pick in the middle subdomain is greater than $1/2$ on each recursive call of **LazySample**, we expect at most $5 \log M$ recursive calls to reach domain size $M < 12$. Therefore, in total at most $5 \log M + 12$ recursive calls are needed on average to map an input domain point. ∎

Note that the algorithms make one call to HGD on each recursion, so an upper-bound on their running-times is then at most $(\log N + 1) \cdot T_{\text{HGD}}$ in the worst-case and at most $(5 \log M + 12) \cdot T_{\text{HGD}}$ on average, where T_{HGD} denotes the running-time of HGD on inputs of size at most $\log N$. However, this does not take into account the fact that the size of these inputs decrease on each recursion. Thus, better bounds may be obtained by analyzing the running-time of a specific realization of HGD.

4.5 Realizing HGD

An efficient implementation of sampling algorithm HGD was designed by Kachitvichyanukul and Schmeiser [22]. Their algorithm is exact; it is not an approximation by a related distribution. It is implemented in Wolfram Mathematica and other libraries, and is fast even for large parameters. However, on small parameters the algorithms of [29] perform better. Since the parameter size to HGD in our **LazySample** algorithms shrinks across the recursive calls from large to small, it could be advantageous to switch algorithms at some threshold. We refer the reader to [29, 22, 23, 14] for more details.

We comment that the algorithms of [22] are technically only "exact" when the underlying floating-point operations can be performed to infinite precision. In practice, one has to be careful of truncation error. For simplicity, Theorem 4.2 did not take this into account, as in theory the error can be made arbitrarily small by increasing the precision of floating-point operations (independently of M, N). But we make this point explicit in Theorem 5.3 that analyzes security of our actual scheme.

5 Our OPE Scheme and its Analysis

Algorithms **LazySample**, **LazySampleInv** cannot be directly converted into encryption and decryption procedures because they share and update a joint state, namely arrays F and I, which

store the outputs of the randomized algorithm HGD. For our actual scheme, we can eliminate this shared state by implementing the subroutine GetCoins, which produces coins for HGD, as a PRF and (re-)constructing entries of F and I on-the-fly as needed. However, coming up with a practical yet provably secure construction requires some care. Below we give the details of our PRF implementation for this purpose, which we call TapeGen.

5.1 The TapeGen PRF

Length-Flexible PRFs. In practice, it is desirable that TapeGen be both variable input-length (VIL)- and variable output-length (VOL)-PRF,[2] a primitive we call a *length-flexible* (LF)-PRF. (In particular, the number of coins used by HGD can be beyond one block of an underlying blockcipher in length, ruling out the use of most practical pseudorandom VIL-MACs.) That is, LF-PRF TapeGen with key-space *Keys* takes as input a key $K \in Keys$, an output length 1^ℓ, and $x \in \{0,1\}^*$ to return $y \in \{0,1\}^\ell$. Define the following oracle R taking inputs 1^ℓ and $x \in \{0,1\}^*$ to return $y \in \{0,1\}^\ell$, which maintains as state an array D:

Oracle $R(1^\ell, x)$
  If $|D[x]| < \ell$ then
    $r \xleftarrow{\$} \{0,1\}^{\ell - |D[x]|}$
    $D[x] \leftarrow D[x] \| r$
  Return $D[x]_1 \ldots D[x]_\ell$

Above and in what follows, s_i denotes the i-th bit of a string s, and we require everywhere that $\ell < \ell_{\max}$ for an associated maximum output length $\ell_{\max}$. For an adversary A, define its *lf-prf-advantage* against TapeGen as

$$\mathbf{Adv}^{\text{lf-prf}}_{\mathsf{TapeGen}}(A) \;=\; \Pr[\, A^{\mathsf{TapeGen}(K,\cdot,\cdot)} = 1 \,] - \Pr[\, A^{R(\cdot,\cdot)} = 1 \,] \;,$$

where the left probability is over the random choice of $K \in Keys$. Most practical VIL-MACs (message authentication codes) are PRFs and are therefore VIL-PRFs, but the VOL-PRF requirement does not seem to have been addressed previously. To achieve it we suggest using a VOL-PRG (pseudorandom generator) as well. Let us define the latter.

Variable-output-length PRGs. Let G be an algorithm that on input a seed $s \in \{0,1\}^k$ and an output length 1^ℓ returns $y \in \{0,1\}^\ell$. Let $\mathcal{O}_G$ be the oracle that on input 1^ℓ chooses a random seed $s \in \{0,1\}^k$ and returns $G(s,\ell)$, and let S be the oracle that on input 1^ℓ returns a random string $r \in \{0,1\}^\ell$. For an adversary A, define its *vol-prg-advantage* against G as

$$\mathbf{Adv}^{\text{vol-prg}}_{G}(A) \;=\; \Pr[\, A^{\mathcal{O}_G(\cdot)} = 1 \,] - \Pr[\, A^{S(\cdot)} = 1 \,] \;.$$

As before, we require above that $\ell < \ell_{\max}$ for an associated maximum output length $\ell_{\max}$. Call G *consistent* if $\Pr[\, G(s,\ell') = G(s,\ell)_1 \ldots G(s,\ell)_{\ell'} \,] = 1$ for all $\ell' < \ell$, with the probability over the choice of a random seed $s \in \{0,1\}^k$. Most PRGs are consistent due to their "iterated" structure.

Our LF-PRF construction. We propose a general construction of an LF-PRF that composes a VIL-PRF with a consistent VOL-PRG, namely using the output of the former as the seed for the latter. Formally, let F be a VIL-PRF and G be a consistent VOL-PRG, and define the associated pseudorandom tape generation function TapeGen which on inputs $K, 1^\ell, x$ returns $G(1^\ell, F(K,x))$. The following says that TapeGen is indeed an LF-PRF if F is a VIL-PRF and G is a VOL-PRG.

[2] That is, a VIL-PRF takes inputs of varying lengths. A VOL-PRF produces outputs of varying lengths specified by an additional input parameter.

Proposition 5.1 *Let A be an adversary against $\mathsf{TapeGen}$ that makes at most q queries to its oracle of total input length ℓ_{in} and total output length ℓ_{out}. Then there exists an adversary B_1 against F and an adversary B_2 against G such that*

$$\mathbf{Adv}^{\text{lf-prf}}_{\mathsf{TapeGen}}(A) \;\le\; 2 \cdot (\mathbf{Adv}^{\text{prf}}_{F}(B_1) + \mathbf{Adv}^{\text{vol-prg}}_{G}(B_2)) \;.$$

Adversaries B_1, B_2 make at most q queries of total input length ℓ_{in} or total output length ℓ_{out} to their respective oracles and run in the time of A. ▮

Proof: We use a standard hybrid argument, changing the experiment where A has oracle $\mathsf{TapeGen}(K, \cdot, \cdot)$ into one with oracle $\mathcal{O}_R(\cdot, \cdot)$ in two steps. Namely, first change the former oracle to on input ℓ, x output not $G(\ell, F(K, x))$ but $G(\ell, s)$ for a independent random $s \in \{0,1\}^k$. The change in A's advantage is bounded by $\mathbf{Adv}^{\text{prf}}_{F}(B_1)$, where B_1 is the PRF adversary against F that runs A, responding to a query ℓ, x by querying its own oracle with x to receive response y, and then returning $G(\ell, y)$ to A. Next change A's oracle to on input ℓ, x return $\mathcal{O}_R(\ell, x)$. This time the change in A's advantage is bounded by $\mathbf{Adv}^{\text{vol-prg}}_{G}(B_2)$, where B_2 is the VOL-PRG adversary against G that runs A, responding to a query ℓ, x with the response it receives to query ℓ to its own oracle, and the proposition follows. ▮

Concretely, we suggest the following blockcipher-based consistent VOL-PRG for G. Let $E\colon \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^n$ be a blockcipher. Define the associated VOL-PRG $G[E]$ with seed-length k and maximum output length $n \cdot 2^n$, where G[E] on input $s \in \{0,1\}^k$ and 1^ℓ outputs the first ℓ bits of the sequence $E(s, \langle 1 \rangle) \| E(s, \langle 2 \rangle) \| \ldots$ (Here $\langle i \rangle$ denotes the n-bit binary encoding of $i \in \mathbb{N}$.) The following says that $G[E]$ is a consistent VOL-PRG if E is a PRF.

Proposition 5.2 *Let $E\colon \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^n$ be a blockcipher, and let A be an adversary against $G[E]$ making at most q oracle queries whose responses total at most $p \cdot n$ bits. Then there is an adversary B against E such that*

$$\mathbf{Adv}^{\text{vol-prg}}_{G[E]}(A) \;\le\; 2q \cdot \mathbf{Adv}^{\text{prf}}_{E}(B) \;.$$

Adversary B makes at most p queries to its oracle and runs in the time of A. Furthermore, $G[E]$ is consistent. ▮

It is easy to prove the above for a VOL-PRG adversary making 1 query, and then the proposition follows by a standard hybrid argument.

Now, to instantiate the VIL-PRF F in the $\mathsf{TapeGen}$ construction, we suggest OMAC (aka. CMAC) [21], which is also blockcipher-based and introduces no additional assumption. Then the secret-key for $\mathsf{TapeGen}$ consists only of that for OMAC, which in turn consists of just one key for the underlying blockcipher (e.g. AES).

5.2 Our OPE Scheme and its Analysis

The scheme. Let $\mathsf{TapeGen}$ be as above, with key-space *Keys*. Our associated order-preserving encryption scheme $\mathcal{OPE}[\mathsf{TapeGen}] = (\mathcal{K}, \mathcal{E}nc, \mathcal{D}ec)$ is defined as follows. The plaintext and ciphertext-spaces are sets of consecutive integers $\mathcal{D}, \mathcal{R}$, respectively. Algorithm $\mathcal{K}$ returns a random $K \in \mathit{Keys}$. Algorithms $\mathcal{E}nc, \mathcal{D}ec$ are the same as **LazySample**, **LazySampleInv**, respectively, except that HGD

$\mathcal{E}nc_K(\mathcal{D},\mathcal{R},m)$
01 $M \leftarrow |\mathcal{D}|$; $N \leftarrow |\mathcal{R}|$
02 $d \leftarrow \min(\mathcal{D}) - 1$; $r \leftarrow \min(\mathcal{R}) - 1$
03 $y \leftarrow r + \lceil N/2 \rceil$
04 If $|\mathcal{D}| = 1$ then
05 $\quad cc \xleftarrow{\$} \mathsf{TapeGen}(K, 1^{\ell_{\mathcal{R}}}, (\mathcal{D},\mathcal{R},1\|m))$
06 $\quad c \xleftarrow{cc} \mathcal{R}$
07 $\quad$ Return c

08 $cc \xleftarrow{\$} \mathsf{TapeGen}(K, 1^{\ell_1}, (\mathcal{D},\mathcal{R},0\|y))$
09 $x \xleftarrow{\$} \mathrm{HGD}(\mathcal{D},\mathcal{R},y;cc)$
10 If $m \le x$ then
11 $\quad \mathcal{D} \leftarrow \{d+1,\ldots,x\}$
12 $\quad \mathcal{R} \leftarrow \{r+1,\ldots,y\}$
13 Else
14 $\quad \mathcal{D} \leftarrow \{x+1,\ldots,d+M\}$
15 $\quad \mathcal{R} \leftarrow \{y+1,\ldots,r+N\}$
16 Return $\mathcal{E}nc_K(\mathcal{D},\mathcal{R},m)$

$\mathcal{D}ec_K(\mathcal{D},\mathcal{R},c)$
17 $M \leftarrow |\mathcal{D}|$; $N \leftarrow |\mathcal{R}|$
18 $d \leftarrow \min(\mathcal{D}) - 1$; $r \leftarrow \min(\mathcal{R}) - 1$
19 $y \leftarrow r + \lceil N/2 \rceil$
20 If $|\mathcal{D}| = 1$ then $m \leftarrow \min(\mathcal{D})$
21 $\quad cc \xleftarrow{\$} \mathsf{TapeGen}(K, 1^{\ell_{\mathcal{R}}}, (\mathcal{D},\mathcal{R},1\|m))$
22 $\quad w \xleftarrow{cc} \mathcal{R}$
23 $\quad$ If $w = c$ then return m
24 $\quad$ Else return $\perp$
25 $cc \xleftarrow{\$} \mathsf{TapeGen}(K, 1^{\ell_1}, (\mathcal{D},\mathcal{R},0\|y))$
26 $x \xleftarrow{\$} \mathrm{HGD}(\mathcal{D},\mathcal{R},y;cc)$
27 If $c \le y$ then
28 $\quad \mathcal{D} \leftarrow \{d+1,\ldots,x\}$
29 $\quad \mathcal{R} \leftarrow \{r+1,\ldots,y\}$
30 Else
31 $\quad \mathcal{D} \leftarrow \{x+1,\ldots,d+M\}$
32 $\quad \mathcal{R} \leftarrow \{y+1,\ldots,r+N\}$
33 Return $\mathcal{D}ec_K(\mathcal{D},\mathcal{R},c)$

Figure 2: The $\mathcal{E}nc$, $\mathcal{D}ec$ algorithms.

is implemented by the algorithm of [22] and GetCoins by $\mathsf{TapeGen}$ (so there is no need to store the elements of F and I). See Figure 2 for the formal descriptions of $\mathcal{E}nc$ and $\mathcal{D}ec$, where as before $\ell_1 = \ell(\mathcal{D},\mathcal{R},y)$ is the number of coins needed by HGD on inputs $\mathcal{D},\mathcal{R},y$, and $\ell_{\mathcal{R}}$ is the number of coins needed to select an element of $\mathcal{R}$ uniformly at random. (The length parameters to $\mathsf{TapeGen}$ are just for convenience; one can always generate more output bits on-the-fly by invoking $\mathsf{TapeGen}$ again on a longer such parameter. In fact, our implementation of $\mathsf{TapeGen}$ can simply pick up where it left off instead of starting over.)

Security. The following theorem characterizes security of our OPE scheme, saying that it is POPF-CCA secure if $\mathsf{TapeGen}$ is a LF-PRF. Applying Proposition 5.2, this is reduced to pseudorandomness of an underlying blockcipher.

Theorem 5.3 *Let $\mathcal{OPE}[\mathsf{TapeGen}]$ be the OPE scheme defined above with plaintext-space of size M and ciphertext-space of size N. Then for any adversary A against $\mathcal{OPE}[\mathsf{TapeGen}]$ making at most q queries to its oracles combined, there is an adversary B against $\mathsf{TapeGen}$ such that*

$$\mathbf{Adv}^{\text{popf-cca}}_{\mathcal{OPE}[\mathsf{TapeGen}]}(A) \;\le\; \mathbf{Adv}^{\text{lf-prf}}_{\mathsf{TapeGen}}(B) \;+\; \lambda\,.$$

Adversary B makes at most $q_1 = q \cdot (\log N + 1)$ queries of size at most $5 \log N + 1$ to its oracle, whose responses total $q_1 \cdot \lambda'$ bits on average, and its running-time is that of A. Above, λ, λ' are constants depending only on HGD and the precision of the underlying floating-point computations (not on M, N). ∎

Proof:

$$\begin{aligned}
\mathbf{Adv}^{\text{popf-cca}}_{\mathcal{OPE}[\mathsf{TapeGen}]}(A) &= \Pr[\, A^{\mathcal{E}nc(K,\cdot),\mathcal{D}ec(K,\cdot)} = 1\,] - \Pr[\, A^{g(\cdot),g^{-1}(\cdot)} = 1\,] \\
&= \Pr[\, A^{\mathcal{E}nc(K,\cdot),\mathcal{D}ec(K,\cdot)} = 1\,] - \Pr[\, A^{\mathbf{LazySample}(\mathcal{D},\mathcal{R},\cdot),\mathbf{LazySampleInv}(\mathcal{D},\mathcal{R},\cdot)} = 1\,] \\
&\leq \mathbf{Adv}^{\text{lf-prf}}_{\mathsf{TapeGen}}(B) \;+\; \lambda\,.
\end{aligned}$$

The first equation is by definition. The second equation is due to Theorem 4.2. The last inequality is justified as follows. Adversary B is given an oracle for either TapeGen or a random function with corresponding inputs and outputs lengths. It runs A and replies to its oracle queries by simulating $\mathcal{E}nc$ and $\mathcal{D}ec$ algorithms. Note that only the procedure TapeGen used by these algorithms uses the secret key. B simulates it using its own oracle. By construction our $\mathcal{E}nc$ and $\mathcal{D}ec$ algorithms differ from **LazySample** and **LazySampleInv** respectively only in the use of random tape, which is truly random in one case and pseudorandom in another. Thus any difference in the probabilities in the second line will result the difference B's output distribution which is $\mathbf{Adv}^{\text{prf}}_{\mathsf{TapeGen}}(B)$. Above λ represents an "error term" due to the fact that the "exact" hypergeometric sampling algorithm of [22] technically requires infinite floating-point precision, which is not possible in the real world. One way to bound λ would be to bound the probability that an adversary can distinguish the used HGD sampling algorithm from the ideal (infinite precision) one. B's running time and resources are justified by observing the algorithms and their efficiency analysis. ▌

Efficiency. The efficiency of our scheme follows from our previous analyses. Using the suggested implementation of TapeGen in Subsection 5.1, encryption and decryption require the time for at most $\log N + 1$ invocations of HGD on inputs of size at most $\log N$ plus at most $(5 \log M + 12) \cdot (5 \log N + \lambda' + 1)/128$ invocations of AES on average for λ' in the theorem.

5.3 On Choosing N

One way to choose the size of the ciphertext-space N for our scheme is just to ensure the number of functions $[M]$ to $[N]$ is very large, say more than 2^{80}. (We assume that the size of the plaintext-space M is given.) The number of such functions, which is given by $\binom{N}{M}$, is maximized when $M = N/2$. And, since $(N/M)^M \leq \binom{N}{M}$, it is greater than 2^{80} as long as $M = N/2 > 80$. However, once we have a greater understanding of what information about the data is leaked by a random order-preserving function (the "ideal object" in our POPF-CCA definition), more sophisticated criteria might be used to select N. In fact, it would also be possible to view our scheme more as a "tool" like a blockcipher rather than a full-fledged encryption scheme itself, and to try to use it to design an OPE scheme with better security in some cases. We leave these as interesting and important directions for future work.

6 On Using the Negative Hypergeometric Distribution

In the balls-and-bins model described in Section 4.1 with M black and $N - M$ white balls in the bin, consider the random variable Y describing the total number of balls in our sample after we pick the x-th black ball. This random variable follows the *negative* hypergeometric (NHG) distribution. Formally,

$$P_{NHGD}(y; N, M, x) = \frac{\binom{y-1}{x-1} \cdot \binom{N-y}{M-x}}{\binom{N}{M}}.$$

As we discussed in the introduction, use of the NHG distribution instead of the HG permits slightly simpler and more efficient lazy sampling algorithms and corresponding OPE scheme. The problem is that they require an efficient NHG sampling algorithm, and the existence of such an algorithm is apparently open. What is known is that the NHG distribution can be approximated by the negative binomial distribution [26], the latter can be sampled efficiently [16, 14], and the approximation improves as M and N grow. However, quantifying the quality of the approximation for fixed parameters seems difficult. If future work either develops an efficient exact sampling algorithm for the NHG distribution or shows that the approximation by the negative binomial distribution is sufficiently close, then our NHG-based OPE scheme could be a good alternative to the HG-based one. Here are the details.

6.1 Construction of the NHGD-based OPE Scheme

Assume there exists an efficient algorithm NHGD that efficiently samples according to the NHG distribution, possibly using an approximation to a related distribution as we discussed. NHGD takes inputs $\mathcal{D}, \mathcal{R}$, and $x \in \mathcal{D}$ and returns $y \in \mathcal{R}$ such that for each $y^* \in \mathcal{R}$ we have $y = y^*$ with probability $P_{NHGD}(y - r; |\mathcal{R}|, |\mathcal{D}|, x - d)$ over the coins of NHGD, where $d = \min(\mathcal{D}) - 1$ and $r = \min(\mathcal{R}) - 1$. Let $\ell_1 = \ell(\mathcal{D}, \mathcal{R}, y)$ denote the number of coins needed by NHGD on inputs $\mathcal{D}, \mathcal{R}, x$.

The NHGD-based order-preserving encryption scheme $\mathcal{OPE}[\mathsf{TapeGen}] = (\mathcal{K}, \mathcal{E}nc^\star, \mathcal{D}ec^\star)$ is defined as follows. Let $\mathsf{TapeGen}$ be the PRF described in Section 5, with key-space *Keys*. The plaintext and ciphertext-spaces are sets of consecutive integers $\mathcal{D}, \mathcal{R}$, respectively. Algorithm $\mathcal{K}$ returns a random $K \in \mathit{Keys}$. Algorithms $\mathcal{E}nc^\star, \mathcal{D}ec^\star$ are described in Figure 3.

$\mathcal{E}nc^\star_K(\mathcal{D}, \mathcal{R}, m)$	$\mathcal{D}ec^\star_K(\mathcal{D}, \mathcal{R}, c)$
	15 If $\lvert\mathcal{D}\rvert = 0$ then return $\perp$
01 $M \leftarrow \lvert\mathcal{D}\rvert$; $N \leftarrow \lvert\mathcal{R}\rvert$	16 $M \leftarrow \lvert\mathcal{D}\rvert$; $N \leftarrow \lvert\mathcal{R}\rvert$
02 $d \leftarrow \min(\mathcal{D}) - 1$; $r \leftarrow \min(\mathcal{R}) - 1$	17 $d \leftarrow \min(\mathcal{D}) - 1$; $r \leftarrow \min(\mathcal{R}) - 1$
03 $x \leftarrow d + \lceil M/2 \rceil$	18 $x \leftarrow d + \lceil M/2 \rceil$
04 $cc \xleftarrow{\$} \mathsf{TapeGen}(K, 1^{\ell_1}, (\mathcal{D}, \mathcal{R}, x))$	19 $cc \xleftarrow{\$} \mathsf{TapeGen}(K, 1^{\ell_1}, (\mathcal{D}, \mathcal{R}, x))$
05 $y \leftarrow \mathrm{NHGD}(\mathcal{R}, \mathcal{D}, x; cc)$	20 $y \leftarrow \mathrm{NHGD}(\mathcal{R}, \mathcal{D}, x; cc)$
06 If $m = x$ then	21 If $c = y$ then
07 Return y	22 Return x
08 If $m < x$ then	23 If $c < y$ then
09 $\mathcal{D} \leftarrow \{d + 1, \ldots, x - 1\}$	24 $\mathcal{D} \leftarrow \{d + 1, \ldots, x - 1\}$
10 $\mathcal{R} \leftarrow \{r + 1, \ldots, y - 1\}$	25 $\mathcal{R} \leftarrow \{r + 1, \ldots, y - 1\}$
11 Else	26 Else
12 $\mathcal{D} \leftarrow \{x + 1, \ldots, d + M\}$	27 $\mathcal{D} \leftarrow \{x + 1, \ldots, d + M\}$
13 $\mathcal{R} \leftarrow \{y + 1, \ldots, r + N\}$	28 $\mathcal{R} \leftarrow \{y + 1, \ldots, r + N\}$
14 Return $\mathcal{E}nc_K(\mathcal{D}, \mathcal{R}, m)$	29 Return $\mathcal{D}ec_K(\mathcal{D}, \mathcal{R}, c)$

Figure 3: The $\mathcal{E}nc^\star$, $\mathcal{D}ec^\star$ algorithms for the NHGD scheme.

6.2 Correctness

We prove correctness of the NHGD scheme in the same manner as the HGD scheme. First, consider the following revised versions of stateful algorithms **LazySample***, **LazySampleInv***. The algorithms

use the subroutine GetCoins from before, which takes inputs 1^ℓ,$\mathcal{D}$,$\mathcal{R}$, and $b\|z$, where $b \in \{0,1\}$ and $z \in \mathcal{R}$ if $b = 0$ and $z \in \mathcal{D}$ otherwise, to return $cc \in \{0,1\}^\ell$. Also, recall that the array I, initially empty, is global and shared between the algorithms. The algorithm descriptions are given in Figure 4.

LazySample$^\star(\mathcal{D}, \mathcal{R}, m)$

$M \leftarrow |\mathcal{D}|$; $N \leftarrow |\mathcal{R}|$
$d \leftarrow \min(\mathcal{D}) - 1$; $r \leftarrow \min(\mathcal{R}) - 1$
$x \leftarrow d + \lceil M/2 \rceil$
If $I[\mathcal{D}, \mathcal{R}, x]$ is undefined then
    $cc \xleftarrow{\$} \text{GetCoins}(1^{\ell_1}, \mathcal{D}, \mathcal{R}, 0\|x)$
    $I[\mathcal{D}, \mathcal{R}, x] \xleftarrow{\$} \text{NHGD}(\mathcal{D}, \mathcal{R}, x; cc)$
$y \leftarrow I[\mathcal{D}, \mathcal{R}, x]$
If $m = x$ then
    Return y
If $m < x$ then
    $\mathcal{D} \leftarrow \{d+1, \ldots, x-1\}$
    $\mathcal{R} \leftarrow \{r+1, \ldots, y-1\}$
Else
    $\mathcal{D} \leftarrow \{x+1, \ldots, d+M\}$
    $\mathcal{R} \leftarrow \{y+1, \ldots, r+N\}$
Return **LazySample**$^\star(\mathcal{D}, \mathcal{R}, m)$

LazySampleInv$^\star(\mathcal{D}, \mathcal{R}, c)$

If $|\mathcal{D}| = 0$ then return $\perp$
$M \leftarrow |\mathcal{D}|$; $N \leftarrow |\mathcal{R}|$
$d \leftarrow \min(\mathcal{D}) - 1$; $r \leftarrow \min(\mathcal{R}) - 1$
$x \leftarrow d + \lceil M/2 \rceil$
If $I[\mathcal{D}, \mathcal{R}, x]$ is undefined then
    $cc \xleftarrow{\$} \text{GetCoins}(1^{\ell_1}, \mathcal{D}, \mathcal{R}, 0\|x)$
    $I[\mathcal{D}, \mathcal{R}, x] \xleftarrow{\$} \text{NHGD}(\mathcal{D}, \mathcal{R}, x; cc)$
$y \leftarrow I[\mathcal{D}, \mathcal{R}, x]$
If $c = y$ then
    Return x
If $c < y$ then
    $\mathcal{D} \leftarrow \{d+1, \ldots, x-1\}$
    $\mathcal{R} \leftarrow \{r+1, \ldots, y-1\}$
Else
    $\mathcal{D} \leftarrow \{x+1, \ldots, d+M\}$
    $\mathcal{R} \leftarrow \{y+1, \ldots, r+N\}$
Return **LazySampleInv**$^\star(\mathcal{D}, \mathcal{R}, c)$

Figure 4: The revised **LazySample***, **LazySampleInv*** algorithms for the NHGD scheme.

With these revised versions of **LazySample***, **LazySampleInv***, we supply a revised version of Theorem 4.2 for the NHGD case.

Theorem 6.1 *Suppose GetCoins returns truly random coins on each new input. Then for any (even computationally unbounded) algorithm A we have*

$$\Pr[\, A^{g(\cdot), g^{-1}(\cdot)} = 1 \,] \;=\; \Pr[\, A^{\textbf{LazySample}^\star(\mathcal{D},\mathcal{R},\cdot), \textbf{LazySampleInv}^\star(\mathcal{D},\mathcal{R},\cdot)} = 1 \,] \,,$$

where g, g^{-1} denote an order-preserving function picked at random from $\mathsf{OPF}_{\mathcal{D},\mathcal{R}}$ and its inverse, respectively. ∎

Proof: Since we consider unbounded adversaries, we can ignore the inverse oracle in our analysis, since such an adversary can always query all points in the domain to learn all points in the image. Let $M = |\mathcal{D}|$, $N = |\mathcal{R}|$, $d = \min(\mathcal{D}) - 1$, and $r = \min(\mathcal{R}) - 1$. We will say that two functions $g, h : \mathcal{D} \to \mathcal{R}$ are *equivalent* if $g(m) = h(m)$ for all $m \in \mathcal{D}$. (Note that if $\mathcal{D} = \emptyset$, any two functions $g, h : \mathcal{D} \to \mathcal{R}$ are vacuously equivalent.) Let f be any function in $\mathsf{OPF}_{\mathcal{D},\mathcal{R}}$. To prove the theorem, it is enough to show that the function defined by **LazySample**$^\star(\mathcal{D}, \mathcal{R}, \cdot)$ is equivalent to f with probability $1/|\mathsf{OPF}_{\mathcal{D},\mathcal{R}}|$. We prove this using strong induction on M and N.

Consider the base case where $M = 1$, i.e., $\mathcal{D} = \{m\}$ for some m, and $N \geq M$. When it is first called, **LazySample**$^\star(\mathcal{D}, \mathcal{R}, m)$ will determine random coins cc, then enter the result of $\text{NHGD}(\mathcal{D}, \mathcal{R}, m; cc)$ into $I[\mathcal{D}, \mathcal{R}, m]$, whereupon this any future calls of **LazySample**$^\star(\mathcal{D}, \mathcal{R}, m)$ will

always output $F[\mathcal{D},\mathcal{R},m]=c$. Note that by definition, $\mathrm{NHGD}(\mathcal{D},\mathcal{R},m;cc)$ returns $f(m)$ with probability

$$P_{NHGD}(f(m)-r;|\mathcal{R}|,1,1)=\frac{\binom{f(m)-r-1}{0}\cdot\binom{(N-r)-(f(m)-r)}{0}}{\binom{N-r}{1}}=\frac{1}{N-r}=\frac{1}{|\mathcal{R}|}.$$

Thus, the output of $\mathbf{LazySample}^{\star}(\mathcal{D},\mathcal{R},m)$ will always be $f(m)$ with probability $1/|\mathcal{R}|$, implying that $\mathbf{LazySample}^{\star}(\mathcal{D},\mathcal{R},m)$ is equivalent to $f(m)$ with probability $1/|\mathcal{R}|=1/|\mathsf{OPF}_{\mathcal{D},\mathcal{R}}|$.

Now suppose $M>1$, and $N\geq M$. As an induction hypothesis assume that for all domains $\mathcal{D}'$ of size M' and ranges $\mathcal{R}'$ of size $N'\geq M'$, where either $M'<M$ or ($M'=M$ and $N'<N$), and for any function f' in $\mathsf{OPF}_{\mathcal{D}',\mathcal{R}'}$, $\mathbf{LazySample}^{\star}(\mathcal{D}',\mathcal{R}',\cdot)$ is equivalent to f' with probability $1/|\mathsf{OPF}_{\mathcal{D}',\mathcal{R}'}|$.

The first time it is called, $\mathbf{LazySample}^{\star}(\mathcal{D},\mathcal{R},\cdot)$ first computes $I[\mathcal{D},\mathcal{R},x]\xleftarrow{\$}\mathrm{NHGD}(\mathcal{R},\mathcal{D},x;cc)$, where $x=d+\lceil M/2\rceil$. Henceforth, on this and future calls of $\mathbf{LazySample}^{\star}(\mathcal{D},\mathcal{R},\cdot)$, the algorithm sets $y\leftarrow I[\mathcal{D},\mathcal{R},x]$, and follow one of three routes: if $x=m$, the algorithm terminates and returns y, if $m<x$ it will return the output of $\mathbf{LazySample}^{\star}(\mathcal{D}_1,\mathcal{R}_1,m)$, and if if $m>x$ it will return the output of $\mathbf{LazySample}^{\star}(\mathcal{D}_2,\mathcal{R}_2,m)$, where $\mathcal{D}_1=\{1,\ldots,x-1\}$, $\mathcal{R}_1=\{1,\ldots,y-1\}$, $\mathcal{D}_2=\{x+1,\ldots,M\}$, $\mathcal{R}_2=\{y+1,\ldots,N\}$. Let f_1 be f restricted to the domain $\mathcal{D}_1$, and let f_2 be f restricted to the domain $\mathcal{D}_2$. Note then that $\mathbf{LazySample}^{\star}(\mathcal{D},\mathcal{R},\cdot)$ is equivalent to f if and only if all three of the following events occur:

E_1: The invocation of $\mathrm{NHGD}(\mathcal{R},\mathcal{D},x;cc)$ returns the value $f(x)$.

E_2: $\mathbf{LazySample}^{\star}(\mathcal{D}_1,\mathcal{R}_1,\cdot)$ is equivalent to f_1.

E_3: $\mathbf{LazySample}^{\star}(\mathcal{D}_2,\mathcal{R}_2,\cdot)$ is equivalent to f_2.

By the law of conditional probability, and since E_2 and E_3 are independent,

$$\Pr[\,E_1\cap E_2\cap E_3\,]=\Pr[\,E_1\mid\cdot\,]\Pr[\,E_2\cap E_3\mid E_1\mid=\,]\Pr[\,E_1\,]\cdot\Pr[\,E_2\mid E_1\,]\cdot\Pr[\,E_3\mid E_1\,]\ .$$

$\Pr[\,E_1\,]$ is the negative hypergeometric probability that $\mathrm{HGD}(\mathcal{R},\mathcal{D},y-r)$ will return $f(x)$, which is

$$\Pr[\,E_1\,]=P_{NHGD}(f(x)-r;N,M,\lceil M/2\rceil)=\frac{\binom{f(x)-r-1}{\lceil M/2\rceil-1}\binom{N-f(x)+r}{M-\lceil M/2\rceil)}}{\binom{N}{M}}.$$

Assume that E_1 holds, and thus f_1 is an element of $\mathsf{OPF}_{\mathcal{D}_1,\mathcal{R}_1}$ and f_2 is an element of $\mathsf{OPF}_{\mathcal{D}_2,\mathcal{R}_2}$. By definition, $|\mathcal{R}_1|,|\mathcal{R}_2|<|\mathcal{R}|$, and $|\mathcal{D}_1|,|\mathcal{D}_2|\leq|\mathcal{D}|$. So the induction hypothesis holds for each, and thus $\mathbf{LazySample}^{\star}(\mathcal{D}_1,\mathcal{R}_1,\cdot)$ is equivalent to f_1 with probability $1/|\mathsf{OPF}_{\mathcal{D}_1,\mathcal{R}_1}|=1/\binom{|\mathcal{R}_1|}{|\mathcal{D}_1|}$, and $\mathbf{LazySample}^{\star}(\mathcal{D}_2,\mathcal{R}_2,\cdot)$ is equivalent to f_2 with probability $1/|\mathsf{OPF}_{\mathcal{D}_2,\mathcal{R}_2}|=1/\binom{|\mathcal{R}_2|}{|\mathcal{D}_2|}$. Thus, we have that

$$\Pr[\,E_2\mid E_1\,]=\frac{1}{\binom{f(x)-r-1}{\lceil M/2\rceil-1}}\qquad\text{and}\qquad\Pr[\,E_3\mid E_1\,]=\frac{1}{\binom{N-f(x)+r}{M-\lceil M/2\rceil}}\ .$$

$$\Pr[\,E_1\cap E_2\cap E_3\,]=\frac{\binom{f(x)-r-1}{\lceil M/2\rceil-1}\binom{N-f(x)+r}{M-\lceil M/2\rceil)}}{\binom{N}{M}}\frac{1}{\binom{f(x)-r-1}{\lceil M/2\rceil-1}}\frac{1}{\binom{N-f(x)+r}{M-\lceil M/2\rceil}}=\frac{1}{\binom{N}{M}}.$$

Therefore, $\mathbf{LazySample}^{\star}(\mathcal{D}, \mathcal{R}, \cdot)$ is equivalent to f with probability $1/\binom{N}{M} = 1/|\mathsf{OPF}_{\mathcal{D},\mathcal{R}}|$. Since f was an arbitrary element of $\mathsf{OPF}_{\mathcal{D},\mathcal{R}}$, the result follows. ▮

Now, it is straightforward to prove the formal statement of correctness as before.

Theorem 6.2 *Let $\mathcal{OPE}[\mathsf{TapeGen}]$ be the OPE scheme defined above with plaintext-space of size M and ciphertext-space of size N. Then for any adversary A against $\mathcal{OPE}[\mathsf{TapeGen}]$ making at most q queries to its oracles combined, there is an adversary B against $\mathsf{TapeGen}$ such that*

$$\mathbf{Adv}^{\text{popf-cca}}_{\mathcal{OPE}[\mathsf{TapeGen}]}(A) \;\leq\; \mathbf{Adv}^{\text{lf-prf}}_{\mathsf{TapeGen}}(B) \;+\; \lambda\,.$$

Adversary B makes at most $q_1 = q \cdot (\log N + 1)$ queries of size at most $5 \log N + 1$ to its oracle, whose responses total $q_1 \cdot \lambda'$ bits on average, and its running-time is that of A. Above, λ, λ' are constants depending only on NHGD and the precision of the underlying floating-point computations (not on M, N). ▮

Proof: The proof of this theorem is identical to that of Theorem 5.3, except that it uses Theorem 6.1 as a lemma rather than Theorem 4.2. ▮

6.3 Efficiency of the NHGD Scheme

Efficiency-wise, it is not hard to see that to encrypt a single plaintext, each algorithm performs $\log M + 1$ recursions in the worst-case (as opposed to $\log N + 1$ for the HG-based algorithms), as the algorithm finds the desired plaintext via a binary search over the plaintext space, at each recursion calling NHGD to determine the encryption of the midpoint (defined as the last plaintext in the first half of the current plaintext domain). The expected number of recursions is easily deduced as

$$\frac{1}{M} \cdot \left[(\log M + 1) + \sum_{k=1}^{\log M} 2^{k-1} k \right] .$$

A simple inductive proof shows that this value is between $\log M - 1$ and $\log M$. This falls in line with what we expect from a binary-search strategy, where the expected number of iterations is typically only about 1 fewer than the worst-case number of iterations.

The algorithms of the corresponding OPE scheme can be obtained following the same idea of eliminating state by using a length-flexible PRF as described in Section 5.2. The security statement is the same as that of Theorem 5.3, where the last term now corresponds to the error probability of the NHGD algorithm.

Acknowledgements

We thank Anna Lysyanskaya, Silvio Micali, Leonid Reyzin, Ron Rivest, Phil Rogaway and the anonymous reviewers of Eurocrypt 2009 for helpful comments and references. Alexandra Boldyreva and Adam O'Neill are supported in part by Alexandra's NSF CAREER award 0545659 and NSF Cyber Trust award 0831184. Younho Lee was supported in part by the Korea Research Foundation Grant funded by the Korean Government (MOEHRD) (KRF:2007-357-D00243). Also, he is supported by Professor Mustaque Ahamad through the funding provided by IBM ISS and AT&T.

References

[1] R. Agrawal, J. Kiernan, R. Srikant, and Y. Xu. Order-preserving encryption for numeric data. In *SIGMOD '04*, pp. 563–574. ACM, 2004.

[2] G. Amanatidis, A. Boldyreva, and A. O'Neill. Provably-secure schemes for basic query support in outsourced databases. In *DBSec '07*, pp. 14–30. Springer, 2007.

[3] F. L. Bauer. *Decrypted Secrets: Methods and Maxims of Cryptology.* Springer, 2006.

[4] M. Bellare. New proofs for NMAC and HMAC: Security without collision-resistance. In *CRYPTO '06*, pp. 602–619. Springer, 2006.

[5] M. Bellare, A. Boldyreva, L. R. Knudsen, and C. Namprempre. Online ciphers and the Hash-CBC construction. In *CRYPTO '01*, pp. 292–309. Springer, 2001.

[6] M. Bellare, A. Boldyreva, and A. O'Neill. Deterministic and efficiently searchable encryption. In *CRYPTO '07*, pp. 535–552. Springer, 2007.

[7] M. Bellare, M. Fischlin, A. O'Neill, and T. Ristenpart. Deterministic encryption: Definitional equivalences and constructions without random oracles. In *CRYPTO '08*, pp. 360–378. Springer, 2008.

[8] M. Bellare, T. Kohno, and C. Namprempre. Authenticated encryption in SSH: provably fixing the SSH binary packet protocol. In *CCS '02*, pp. 1–11. ACM Press, 2002.

[9] M. Bellare and P. Rogaway. The security of triple encryption and a framework for code-based game-playing proofs. In *EUROCRYPT '06*, pp. 409–426. Springer, 2006.

[10] A. Boldyreva, S. Fehr, and A. O'Neill. On notions of security for deterministic encryption, and efficient constructions without random oracles. In *CRYPTO '08*, pp. 335–359. Springer, 2008.

[11] D. Boneh and B. Waters. Conjunctive, subset, and range queries on encrypted data. In *TCC '07*, pp. 535–554. Springer, 2007.

[12] A. C. Cem Say and A. Kutsi Nircan. Random generation of monotonic functions for Monte Carlo solution of qualitative differential equations. *Automatica*, 41(5):739-754, 2005.

[13] Z. Erkin, A. Piva, S. Katzenbeisser, R. L. Lagendijk, J. Shokrollahi, G. Neven, and M. Barni. Protection and retrieval of encrypted multimedia content: When cryptography meets signal processing. *EURASIP Journal on Information Security*, 2007, Article ID 78943, 2007.

[14] G. S. Fishman. *Discrete-event simulation : modeling, programming, and analysis.* Springer, 2001.

[15] E. A. Fox, Q. F. Chen, A. M. Daoud, and L. S. Heath. Order-preserving minimal perfect hash functions and information retrieval. *ACM Transactions on Information Systems*, 9(3):281–308, 1991.

[16] J. E. Gentle. *Random Number Generation and Monte Carlo Methods.* Springer, 2003.

[17] O. Goldreich, S. Goldwasser, and S. Micali. How to construct random functions. *Journal of the ACM*, 33(4):792–807, 1986.

[18] O. Goldreich, S. Goldwasser, and A. Nussboim. On the implementation of huge random objects. *FOCS '03*, IEEE, 2003.

[19] L. Granboulan and T. Pornin. Perfect block ciphers with small blocks. In *FSE '07*, pp. 452–465. Springer, 2007.

[20] P. Indyk, R. Motwani, P. Raghavan, and S. Vempala. Locality-preserving hashing in multidimensional spaces. In *STOC '97*, pp.s 618–625. ACM, 1997. ACM.

[21] T. Iwata and K. Kurosawa. OMAC: One-Key CBC MAC. In *FSE '03*, pp. 137–161. Springer, 2003.

[22] V. Kachitvichyanukul and B. W. Schmeiser. Computer generation of hypergeometric random variates. *Journal of Statistical Computation and Simulation*, 22(2):127–145, 1985.

[23] V. Kachitvichyanukul and B. W. Schmeiser. Algorithm 668: H2PEC: sampling from the hypergeometric distribution. *ACM Transactions on Mathematical Software*, 14(4):397–398, 1988.

[24] J. Li and E. Omiecinski. Efficiency and security trade-off in supporting range queries on encrypted databases. In *DBSec '05*, pp. 69–83. Springer, 2005.

[25] N. Linial and O. Sasson. Non-expansive hashing. In *STOC '96*, pp. 509–518. ACM, 1996.

[26] F. López-Blázquez and B. Salamanca Miño. Exact and approximated relations between negative hypergeometric and negative binomial probabilities. *Communications in Statistics. Theory and Methods*, 30(5):957–967, 2001.

[27] P. Rogaway and T. Shrimpton. A provable-security treatment of the key-wrap problem. In *EUROCRYPT '06*, pp. 373–390. Springer, 2006.

[28] E. Shi, J. Bethencourt, T-H. H. Chan, D. Song, and A. Perrig. Multi-dimensional range query over encrypted data. In *Symposium on Security and Privacy '07*, pp. 350–364. IEEE, 2007.

[29] A. J. Walker. An efficient method for generating discrete random variables with general distributions. *ACM Transactions on Mathematical Software*, 3:253–256, 1977.

[30] D. Westhoff, J. Girao, and M. Acharya. Concealed data aggregation for reverse multicast traffic in sensor networks: Encryption, key distribution, and routing adaptation. *IEEE Transactions on Mobile Computing*, 5(10):1417–1431, 2006.

[31] J. Xu, J. Fan, M. H. Ammar, and S. B. Moon. Prefix-preserving IP address anonymization: Measurement-based security evaluation and a new cryptography-based scheme. In *ICNP '02*, pp. 280–289. IEEE, 2002.