Addendum to the paper

"Simple Constructions of Almost k-wise Independent Random Variables"

by N. Alon, O. Goldreich, J. Hastad and R. Peralta

(This journal, Volume 3, No. 3, pages 289–304, 1992)

The constructions presented in the above paper use a finite field which is either $GF(2^m)$ or $GF(p)$ for some prime p. The constructions are presented assuming that one has a representation of the field (i.e., an irreducible polynomial of degree m or the prime p, respectively). Such representations could be found, with overwhelmingly high probability, in probabilistic polynomial-time (in m or $|p|$, respectively). The paper contained some remarks indicating how to achieve this goal using only a linear number of unbiased coin tosses. However, in retrospective we feel that some more details should be given.

For uniformity of exposition, we denote by m the logarithm (to base 2) of the size of the required field. The field representations in both cases can be encoded by strings of length m. Furthermore, in both cases about a $\frac{1}{m}$ fraction of all m-bit long strings are valid representations, and one can efficiently determine whether a string is a valid representation. Hence, selecting a valid representation can be done by selecting candidates at random until a valid one is found. As indicated in the paper, to save on randomness, we use an efficient sampling which in turn uses a construction of a sequence of pairwise independent variables, each uniformly distributed in $\{0,1\}^m$.

The problem which arises is that the standard constructions of such pairwise independent sequences use a field of similar cardinality (i.e., with at least 2^m elements), and hence we need a representation for this field, which brings us to a circular argument. The solution is to use the known pairwise independent constructions in a slightly less straightforward manner.

Specifically, suppose we need to generate a t-long sequence of pairwise independent m-bit strings (e.g., in the above application $t = O(m)$). The idea is to combine $\lceil \frac{m}{\lceil \log_2 t \rceil} \rceil$ *independent* sequences, each of pairwise independent $\lceil \log_2 t \rceil$-bit strings. Namely, each m-bit string in the desired sequence is obtained by concatenating the corresponding $\lceil \log_2 t \rceil$-bit strings of the different $\lceil \frac{m}{\lceil \log_2 t \rceil} \rceil$ sequences. Hence, we will use $\lceil \frac{m}{\lceil \log_2 t \rceil} \rceil \cdot 2\lceil \log_2 t \rceil \approx 2m$ random bits just like in the standard construction. Yet, now we need a representation for a field of cardinality $\approx t = O(m)$, rather than 2^m, and such a representation can be easily found by exhaustive search. An alternative solution is obtained by taking a closer look at the standard construction of a t-long sequence of pairwise independent elements over $GF(p)$ for p prime.

The observation is that the construction remains valid when the ring Z_M is used instead of $GF(p)$, provided that M is relatively prime to all integers up to t. Consequently, instead of looking for an $2m$-bit long prime, we merely need an $2m$-bit long integer M that is relatively prime to all integers up to t. Such an integer M can be (deterministically) constructed in time polynomial in t (e.g., by multiplying all primes in the interval $[t+1, 2t]$).

Returning to the application in the paper, we now address the problem of verifying that a candidate representation is indeed valid. In case of irreducible polynomials, there exists an efficient deterministic algorithm for this purpose. However, for testing primality only *randomized* efficient algorithms are known. Fortunately, these efficient algorithms require only a linear number of coin tosses. For example, Bach's algorithm (cf., STOC87), on input p, uniformly selects a single residue mod p, and proceeds deterministically, guaranteeing error probability bounded by $1/\sqrt{p}$. Alternatively, one can iterate either of the classic algorithms of Rabin and Solovay and Strassen, using in these iterations related sequences of coin tosses generated by a random walk on an expander.

We conclude by stressing that in case we need to generate a large prime, we use an additional sample space to generate the coin tosses required in all the invocations of the primality testing algorithm. Namely, we generate a sequence of candidate primes, $p_1, ..., p_n$, along with a sequence of random strings $r_1, ..., r_n$ (to be used by the primality tester). Each of these sequences is generated independently of the other, using the same (randomness-efficient) scheme outlined in the paper and above. We note that each of the p_i's is uniformly distributed in $\{0,1\}^m$, and similarly each of the r_i's is uniformly distributed in $\{0,1\}^{O(m)}$. Hence, a prime is found with overwhelming probability, and an error occurs with negligible probability. (To bound the error probability, note that each r_i is uniformly distributed independently of the corresponding p_i.)

We thank Aravind Srinivasan for pointing out the need for the above clarifications.