Finding Composite Field Isomorphisms

Exhaustive Search Workflow

Christopher Wood

May 6, 2013

Isomorphims between the fields $GF(2^k)/R(z)$ and $GF((2^n)^m)/(P(x), Q(y))$ can be obtained by checking for additive and multiplicative homomorphisms between these two representations. An exhaustive search requires us to check all tuples $(P(x), Q(y), R(z))$ of irreducible polynomials that could be used to define the respective fields, and for each representation, generate all possible isomorphisms. To accomplish this task, we make use of the following programs:

- polyGen.py - Generate all irreducible polynomials for $GF(2^k)$ and $GF((2^n)^m)$.
- genGen.py - Find all primitive elements, or generators, for the fields $GF(2^k)$ and $GF((2^n)^m)$ defined by the irreducible polynomials $R(z)$ and $(P(x), Q(y))$, respectively.
- isoGen.py - Find all field isomorphisms using an exhaustive check for additition and multiplication homomorphism of a mapping between the field elements.

To generate all irreducible polynomials $R(z)$ of degree 16 and all irreducible polynomials $Q(y)$ of degree 2 we can run the following:

```
python polyGen.py 2 2 16 8 2 0 131071 > polyList
```

The first argument is a mode of operation. The second, third, fourth, and fifth arguments are the field characteristic, degree of $R(z)$, degree of $P(x)$ and degree of $Q(y)$, respectively. The remaining arguments specify lower and upper bounds on the unique polynomials to check. This enables the task of polynomial generation to be split among arbitrarily many jobs whose output can be reduced into polyList at the end.

With an appropriate list of polynomial tuples $(P(x), Q(y), R(z))$ in polyList, we can now run genGen.py to find all of the generators for these respective fields. This is done as follows:

```
python genGen.py polyList 0 0 65536 > polyListWithGens
```

The first argument specifies the input file containing the polynomial triples, the next argument specifies the mode of operation, and the last two arguments, again, specify lower and upper bounds on field elements to check. This enables the output of this task to be partitioned among many jobs and then reduced at the end. In fact, the program mergeGenOut.py performs this task, but it can be run offline (i.e. not on the grid).

With the list of generators for each polynomial triple $(P(x), Q(y), R(z))$ in the file polyListWithGens, we can now exhaustively generate all isomorphic mappings between the two fields $GF(2^16)$ and $GF((2^8)^2)$. To do this, we run the following:

```
python isoGen.py polyListWithGens isoMapOut > sortedMaps
```

The first argument is again the input file of polynomial triples $(P(x), Q(y), R(z))$ with their associated generators, and the second argument is the output file name that will store the isomorphic mappings when they are identified. The output of this program, which is a list of output file names isoMapOut_N, is sorted based on the respective cost of each field isomorphism. This output is redirected to the file sortedMaps so that the least expensive isomorphisms can be studied further.