Probabilistic Model Checking - Practical 2

1

Listing 1: EGL3 Changes

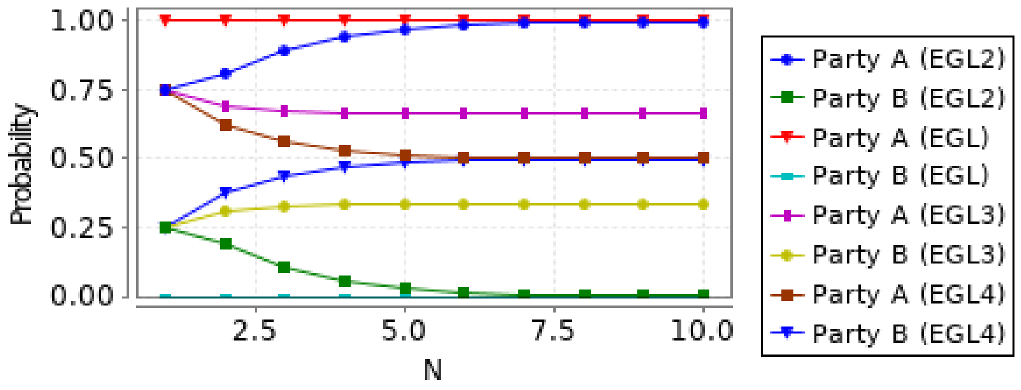
```
// A sends bth bit of nth secret (for n=1..N), move to B
[receiveB] phase=2 & party=1 & n<N -> (party'=2);
// B sends bth bit of nth secret (for n=1..N-1), move to next
    secret and A
[receiveA] phase=2 & party=2 & n<N-1 -> (n'=n+1) & (party'=1);
// B sends bth bit of Nth secret, moves to next bit, 1st secret,
    and back to A
[receiveA] phase=2 & party=2 & n=N-1 & b<L -> (party'=1) & (n'=0)
    & (b'=b+1);
// B sends bth bit of Nth secret, move to next phase (N+1..2N),
    and back to A
[receiveA] phase=2 & party=2 & n=N-1 & b=L -> (party'=1) & (n'=0)
    & (b'=1) & (phase'=3);

// A sends bth bit of nth secret (for n=N+1..2N-1), move to B
[receiveB] phase=3 & party=1 & n<N -> (party'=2);
// B sends bth bit of (N+n)th secret (for n=N+1..2N-1), move to
    next secret and A
[receiveA] phase=3 & party=2 & n<N-1 -> (n'=n+1) & (party'=1);
// B sends bth bit of (N+n)th secret, next bit, 1st secret, and
    back to A
[receiveA] phase=3 & party=2 & n=N-1 & b<L -> (party'=1) & (n'=0)
    & (b'=b+1);
// B sends bth bit of (N+n)th secret, and protocol is now finished.
[receiveA] phase=3 & party=2 & n=N-1 & b=L -> (phase'=4);
```

Listing 2: EGL4 Changes

```
// A sends ith bit of 1st secret to B
[receiveB] phase=2 & party=1 & n=0 -> (party'=2);

// B sends ith bit of (0..N-1)th secret to A
[receiveA] phase=2 & party=2 & n<N-1 -> (n'=n+1);
// B sends (1..L-1)th bit of last secret to A, switches back to A
[receiveA] phase=2 & party=2 & n=N-1 & N>1 -> (n'=1) & (party'=1);
// B sends last bit of the last secret to A, and switches back to A
[receiveA] phase=2 & party=2 & n=N-1 & N=1 & b<L -> (n'=0) &
    (b'=b+1) & (party'=1);
// B sends last bit of the 1st AND last secret to A, next phase
[receiveA] phase=2 & party=2 & n=N-1 & N=1 & b=L -> (n'=0) &
    (b'=1) & (party'=1) & (phase'=3);
// A sends ith bit of (1..N-1)th secret to B
[receiveB] phase=2 & party=1 & n>0 & n<N-1 -> (n'=n+1);
// A sends (1..L-1)th bit of last secret to B, moves to next bit
[receiveB] phase=2 & party=1 & N>1 & n=N-1& b<L-> (n'=0)&(b'=b+1);
// A sends the last bit of last secret to B, moves to next phase
[receiveB] phase=2 & party=1 & N>1 & n=N-1 & b=L -> (n'=0) &
    (b'=1) & (phase'=3);

// A sends ith bit of (N+1)th secret to B
[receiveB] phase=3 & party=1 & n=0 -> (party'=2);

// B sends ith bit of (N..2N-1)th secret to A
[receiveA] phase=3 & party=2 & n<N-1 -> (n'=n+1);
// B sends (1..L-1)th bit of last secret to A, switches back to A
[receiveA] phase=3 & party=2 & n=N-1 & N>1 -> (n'=1) & (party'=1);
// B sends last bit of the last secret to A, and switches back to A
[receiveA] phase=3 & party=2 & n=N-1 & N=1 & b<L -> (n'=0) &
    (b'=b+1) & (party'=1);
// B sends last bit of the 1st AND last secret to A, next phase
[receiveA] phase=3 & party=2 & n=N-1 & N=1 & b=L -> (phase'=4);
// A sends ith bit of (1..N-1)th secret to B
[receiveB] phase=3 & party=1 & n>0 & n<N-1 -> (n'=n+1);
// A sends (1..L-1)th bit of last secret to B, moves to next bit
[receiveB] phase=3 & party=1 & N>1 & n=N-1 & b<L->(n'=0)&(b'=b+1);
// A sends last bit of last secret to B, and moves to next phase
[receiveB] phase=3 & party=1 & N>1 & n=N-1 & b=L -> (phase'=4);
```

2

Figure 1: E[messages until both parties know a pair]



3

Listing 3: Reward structure for E[messages until both parties know a pair]

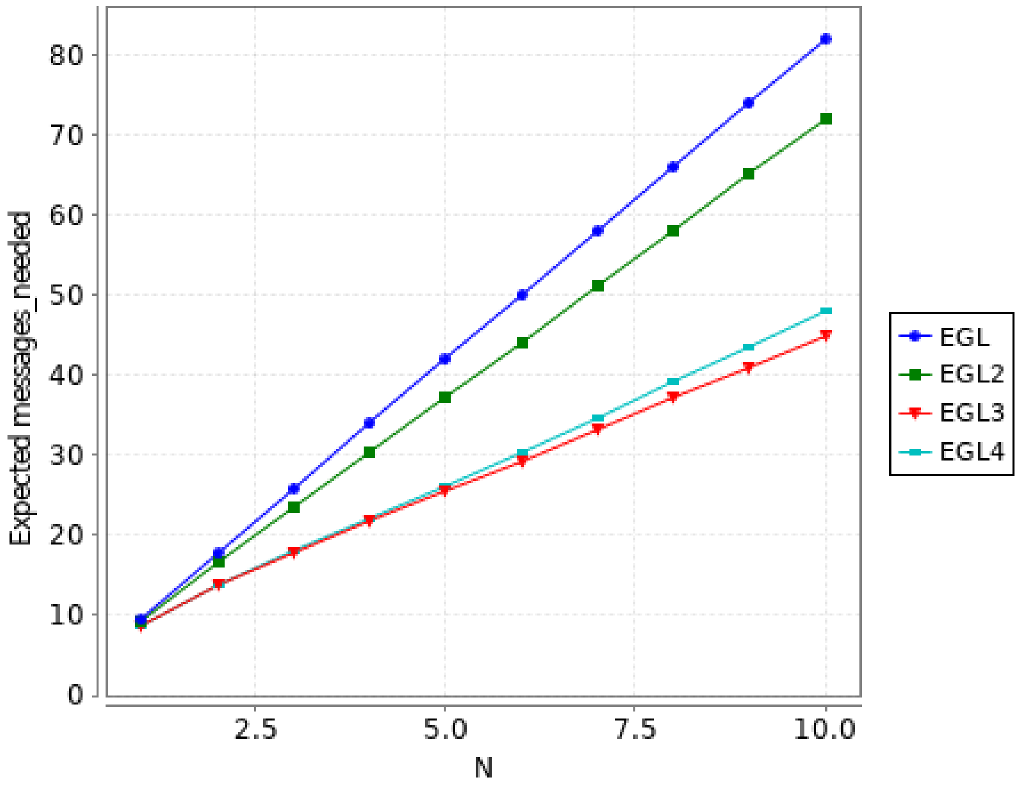
```
rewards "messages_needed"
   [receiveA] !(kB & kA) : 1;
   [receiveB] !(kB & kA) : 1;
endrewards
```

Listing 4: Property for calculating E[messages until both parties know a pair]

```
R{"messages_needed"}=? [ F phase=4 ]
```

4

Figure 2: E[messages until both parties know a pair]


Expected messages_needed
N
EGL
EGL2
EGL3
EGL4