Modeling Multiple Modes of Operation with Alloy (Online Material)

Christos Kalyvas, Elisavet Konstantinou, and Georgios Kambourakis

Laboratory of Information and Communications Systems Security (Info-Sec-Lab),
Department of Information and Communication Systems Engineering,
University of the Aegean, Samos, Greece

ECB|ECB|CBC^{-1}

```
module ECBECBCBC_1
open util/ordering[Block] as ORDER

sig Value { xor : Value one -> one Value }

fact XORfacts {
    all a,b,c : Value | xor.c.(xor.b.a)=xor.(xor.c.b).a
    all b: Value |b.xor =~(b.xor)
    one identity : Value | no identity.xor - iden
}

sig Key { enc: Value one-> one Value }

fact EncFacts {
    all k:Key, b:Value | k.enc[b] !=b
    all disj k,k':Key  | some (k.enc - k'.enc)
}

one sig  K1, K2 extends Key{}

sig Block {
    p: one Value, c1: one Value,iv: one Value,
    c: one Value,  ctemp: one Value
}

fact BlockFacts {
    all b:Block |b.c1 =K1.enc[b.p]
    all b:Block-last| let b' =next[b] | b'.iv=b.c1
    all b:Block |b.ctemp =K2.enc[b.c1]
    all b:Block |b.c =(b.ctemp).xor[b.iv]
}

pred attack {
    all b': Block-first-last | let b=prev[b'], b''=next[b']|
    xor.(b.c1).(b''.c)=xor.(b.iv).(b.c)
}

run attack for exactly 2 Key, exactly 8 Value, exactly 8 Block
```

ECB|OFB|OFB

```
module ECBOFBOFB
open util/ordering[Block] as ORDER

sig Value { xor : Value one -> one Value }

fact XORfacts {
    all a,b,c : Value | xor.c.(xor.b.a)=xor.(xor.c.b).a
    all b: Value |b.xor =~(b.xor)
    one identity : Value | no identity.xor - iden
}

sig Key { enc: Value one-> one Value }

fact EncFacts {
    all k:Key, b:Value | k.enc[b] !=b
    all disj k,k':Key  | some (k.enc - k'.enc)
}

one sig  K,OFB1, OFB2 extends Key{}

sig Block {
    p: one Value, ofb1: one Value, ofb2: one Value,
    pxorofb: one Value, ctemp: one Value, c: one Value
}

fact BlockFacts {
    all b:Block |b.ctemp =K.enc[b.p]
    all b:Block |b.pxorofb =(b.ctemp).xor[b.ofb1]
    all b:Block |b.c =(b.pxorofb).xor[b.ofb2]
    all b:Block-last| let b' =next[b] |
        b'.ofb1=OFB1.enc[b.ofb1] &&
        b'.ofb2= OFB2.enc[b.ofb2]
}

pred attack {
    all b': Block-first-last | let b=prev[b'], b''=next[b'] |
     some d:OFB1.enc.Value | let d'=OFB1.enc[d], d''=OFB1.enc[d']|
     some dd:OFB2.enc.Value | let dd'=OFB2.enc[d], dd''=OFB2.enc[d']|
      xor.d.(xor.dd.(xor.d'.dd'))=xor.(b.c).(b'.c) &&
      xor.d.(xor.dd.(xor.d''.dd''))=xor.(b.c).(b''.c)
}

run attack for exactly 3 Key, exactly 8 Value, exactly 8 Block
```