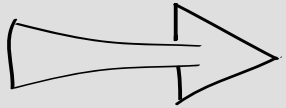
OpenNFP



Protecting The Privacy Of The Network – Using P4 To Prototype And Extend Network Protocols

Mark Matties
JHU Applied Physics Lab
September 7, 2016

Motivation

➡ *Privacy Protection is Incomplete*

We focus solely on end hosts for protecting sensitive information

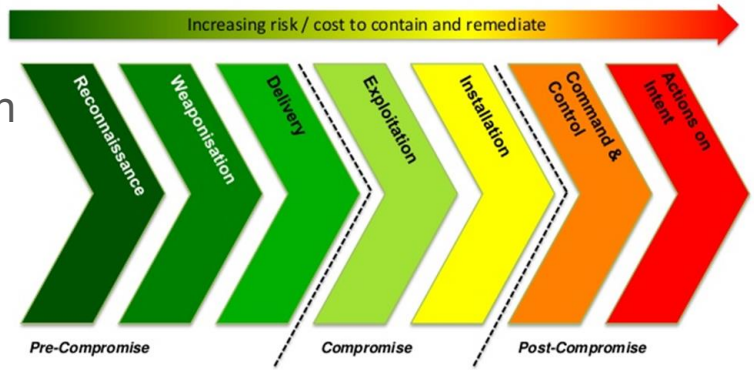
- Encryption of application data at rest and in transit

We ignore protecting sensitive information about the network

- Leak sensitive information about the network useful to adversaries even if application data in transit is encrypted

Reconnaissance is the first step in the cyber kill chain

- Depriving the adversary of information about the network reduces the effectiveness and even likelihood of attack, as well as attack vectors.



Proposed Solution

➡ *Encrypt All Bytes in Transit – Including the Headers*

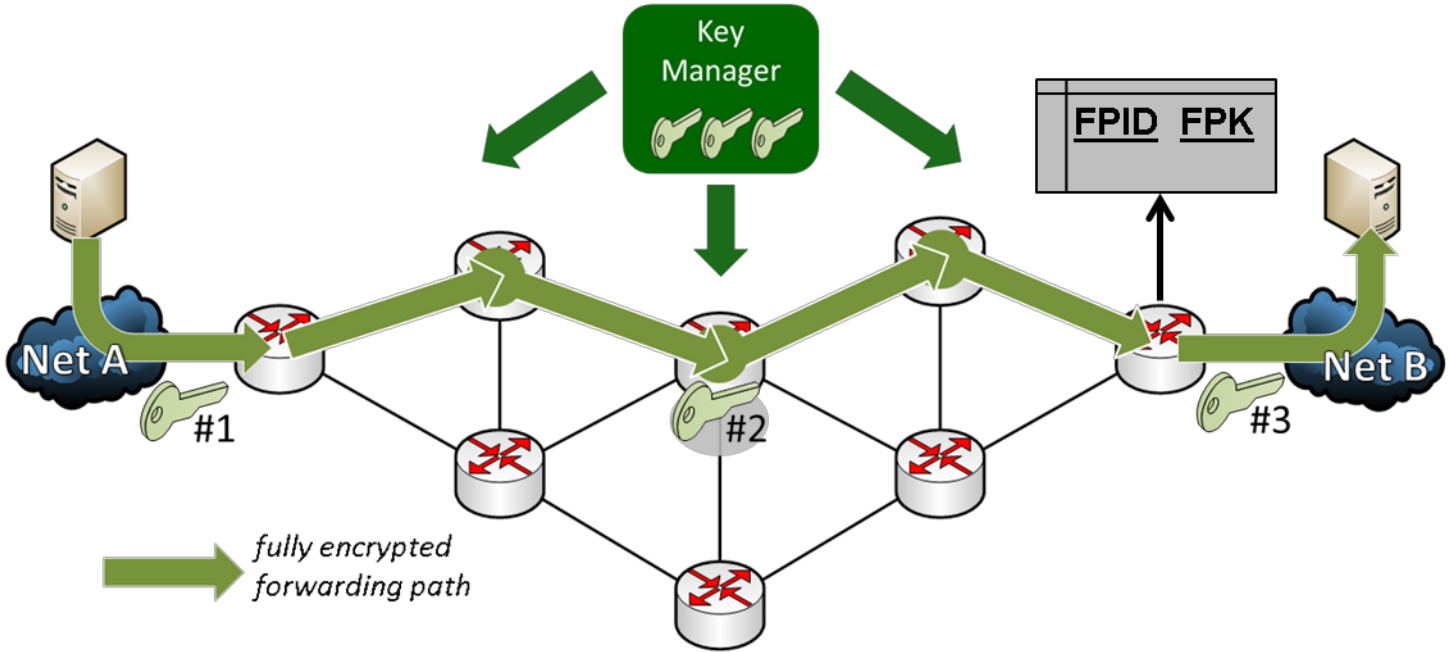
- Can be done independently at Layer 2 and Layer 3
- We're working on both – will describe Layer 3 work today

Solution – Fully Encrypted Network Communications for IP Routing (FENC-IPR)

- Support transmitting and receiving IP datagrams with encrypted headers
  - Requires rethinking of protocols
- Enforce authorized forwarding paths
  - Header encryption prevents unauthorized routers from decrypting the packet header – cannot forward the packet
- Encrypt payload bytes
  - Payload encryption is in not ubiquitous

FENC-IPR serves as a model for other packet forwarding mechanisms

Packets Fully Encrypted End to End

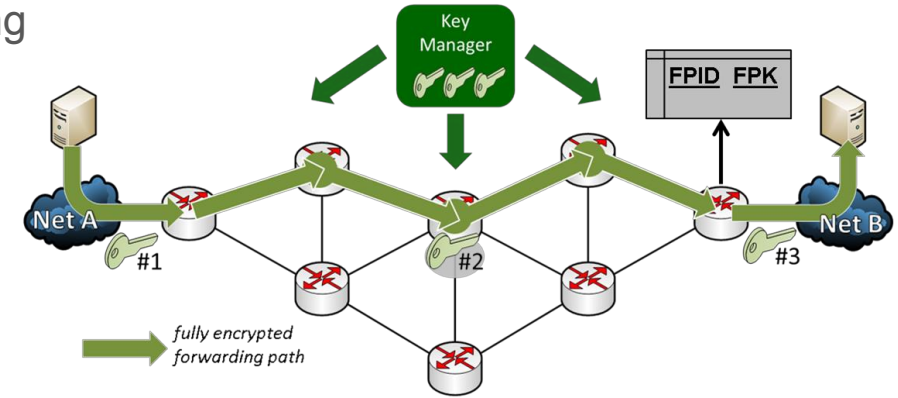


FENC-IPR Components

- Globally unique forwarding path identifier (FPID)
- Logical interface key (LIK) encrypts FENC-IPR header
  - LIKs negotiated by connecting interfaces
- Forwarding path key (FPK) encrypts Layer 3 header
  - FPK associated with unique FPID
  - FPKs distributed by key manager

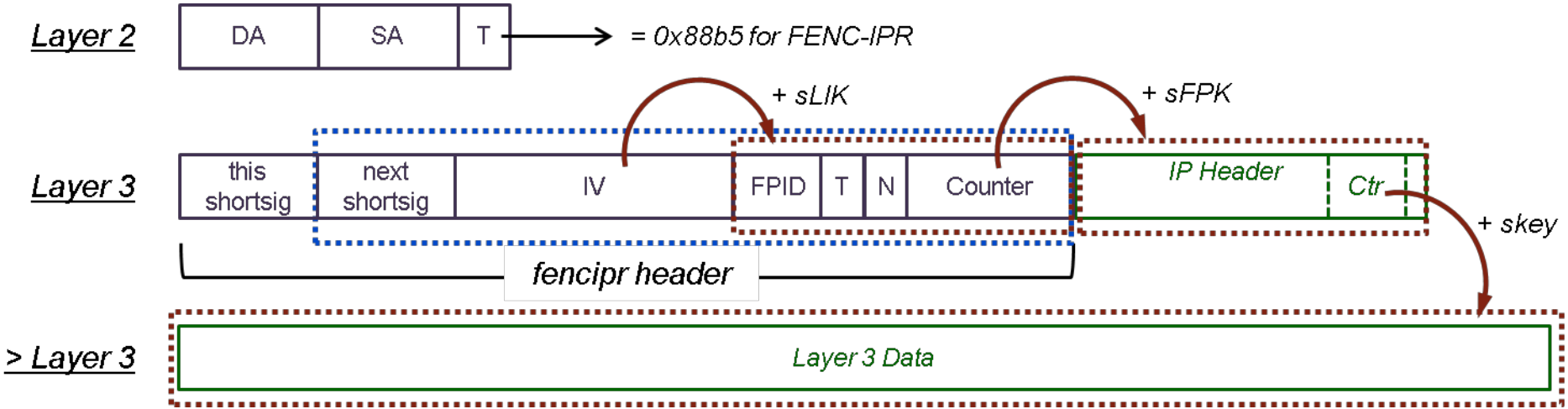
Only routers in the valid forwarding path receive an FPID/FPK pair

- Forwarding path discovery and reporting to key manager is a separate process



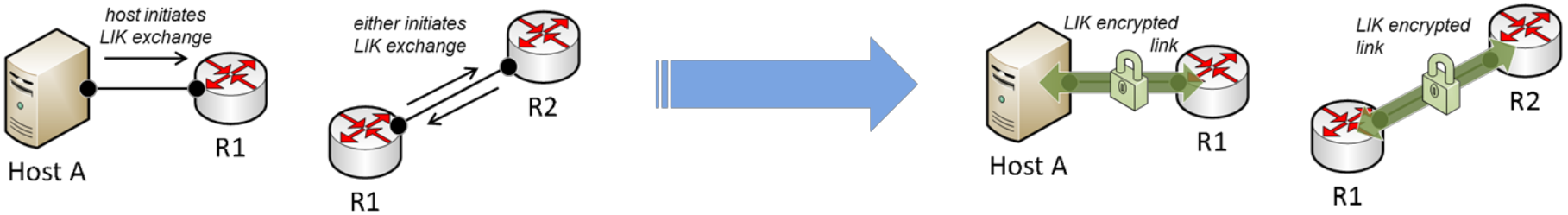
FENC-IPR Components

- Added FENC-IPR header
  - Primary purpose – carry FPID
  - Shortened HMAC digital signatures for quick authentication check
  - Enables support for multiple Layer 3 protocols – IPv4 IPv6 MPLS
- Extended IP header to match encryption block size
  - Carries an optional IV/counter for Layer 3 payload encryption



FENC-IPR – Phases of Operation

1. Upon connection, logical interfaces exchange *logical interface keys* (LIKs)
   - "Upon connection" means configuration of routing table information
   - For encryption domains involving end hosts – the end host initiates LIK exchange with its router.
   - For router only encryption domains, either router of the pair can initiate.
   - "Logical" distinguishes from *physical interface*

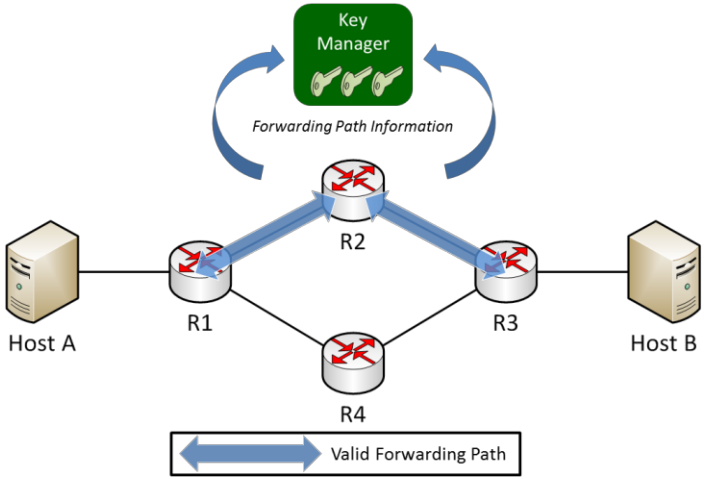


End host / router logical connections also use LIK to encrypt IP header

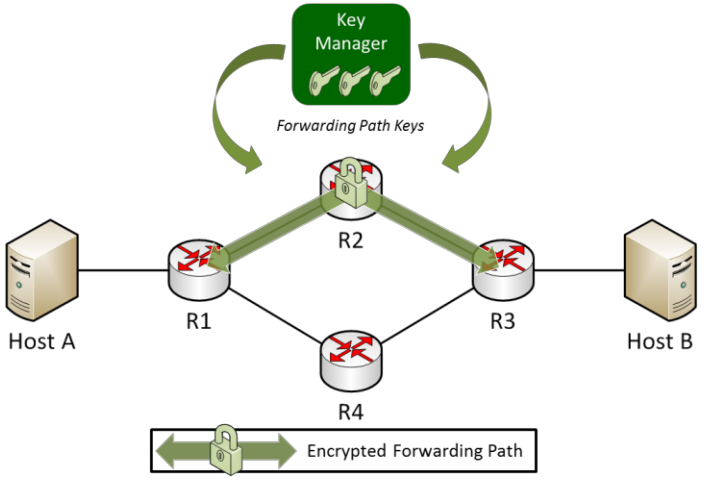
- No separate, additional FPK is used.
- Meant for end hosts that have only a default route / default gateway.
- Reduces burden on Key Manager and maintains a manageable network.

FENC-IPR – Phases of Operation

2. Key Manager collects information on valid forwarding paths

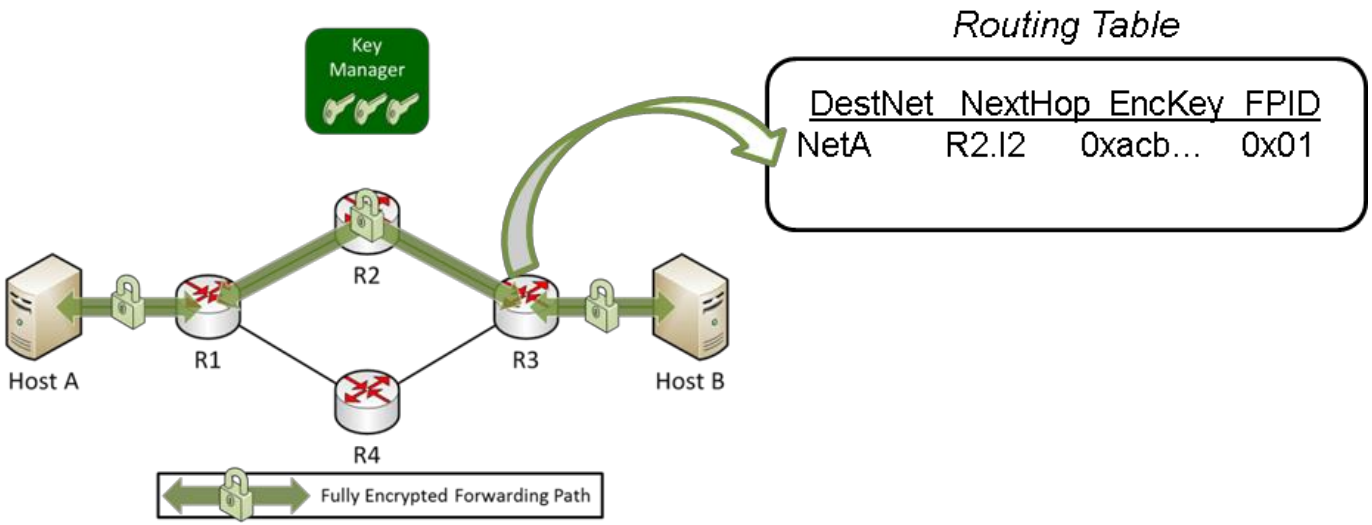


3. Key Manager distributes route key + identifier to routers



FENC-IPR – Phases of Operation

4. Result: Per route fully encrypted forwarding paths established
   - Per interface pair (LIK) encryption of IP header preamble
   - Per route encryption of IP header



FENC-IPR Prototyping

- Concept does not specifically require SDN, but it's easier
- New and modified headers = new protocols

  ➡ *Use P4 to prototype!*
- *P4 benefits for this application (beyond the usual)*
  - *Flexible enough to accommodate headers in packet processing flow that are not part of the packet*
    - *decrypted versions of fencipr and ipv4 headers (inside routers)*
- *P4 gaps for this application*
  - *No primitive actions for encryption/decryption or digital signatures*
  - *Can add them, but it's platform dependent*

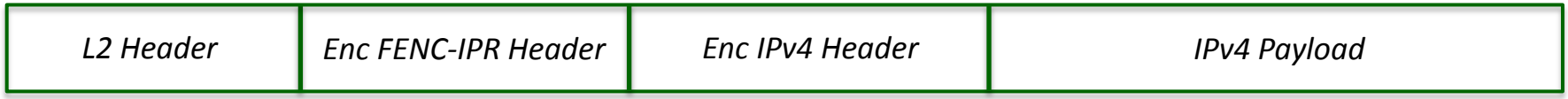
FENC-IPR – P4 Code Samples

- *Packet headers (new)*

```
header_type enc_fencipr_t {
   fields {
        this_shortsig: 64;          // shortened (8B) hmac for a quick check that we encrypted this fencipr header
        next_shortsig: 64;          // shortened (8B) hmac for a quick check that we encrypted the next (L3) header
        this_iv: 128;               // IV needed to decrypt this enc_fencipr_hdr
        enc_bytes : 128;            // the encrypted fpid, etc.
   }
}

header_type enc_layer3_t {
   fields {
        enc_bytes : 256;            // the encrypted Layer 3 header
   }
}
```

Incoming frame



FENC-IPR – P4 Code Samples

- *Internal working headers – not actually sent with packets*

```
header_type fencipr_t {
   fields {
        this_shortsig : 64;        // shortened SHA256 HMAC as a quick sig check for this encrypted header
        next_shortsig : 64;        // shortened SHA256 HMAC as a quick sig check for next encrypted header
        this_iv: 128;              // IV needed to decrypt this enc_fencipr_hdr
        fpid : 32;                 // forwarding path ID (globally unique)
        next_type : 16;            // ethertype of next header ( = 0x0800 for IP)
        next_length : 16;          // length in bytes of next header ( = 32 for IP w/AES_CTR pad)
        aesctr : 64;               // AES counter (8B) used for AES_CTR mode to encrypt Layer 3 header
   }
}
```

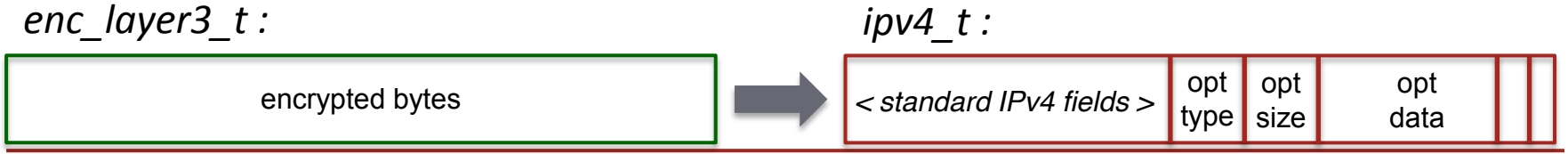
enc_fencipr_t :

fencipr_t :

FENC-IPR – P4 Code Samples

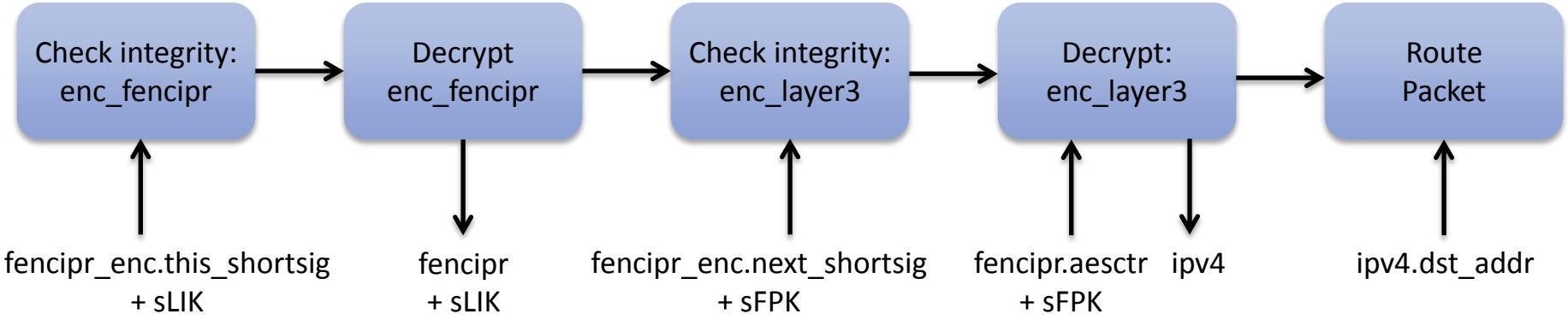
- *Internal working headers – not actually sent with packets*

```
// we use a standard IPv4 header with a "required option" that also pads the length to 32B
header_type ipv4_t {
   fields {
      < std IPv4 header fields >
      // This option is required for FENC-IPR compatible IPv4
      // AES_CBC mode: add 12 bytes to pad to 32 bytes = 2 x 16 bytes (size of AES128 block)
      opt_aesctr_type : 8;          // type as in the IPv4 ID for this option, = 0x1e (expt'l value of 30)
      opt_aesctr_size : 8;          // size of this option in bytes, not including the type field, = 0x0b
      opt_aesctr_data : 64;         // AES_CTR used to encrypt the Layer 3 payload (data)
      opt_eol_type : 8;             // "EOL" = IPv4 end of option list (terminator) = 0x01
      opt_eol_length : 8;           // size of this option in bytes, not including the type field = 0x01
   }
}
```



FENC-IPR – P4 Processing Flow

- *Ingress Control*



FENC-IPR – P4 Code Samples

- *Ingress control logic*

```
control ingress {
    apply(ingress_check_fencipr);
    apply(decrypt_fencipr);
    apply(ingress_check_layer3);
    apply(decrypt_layer3);
    apply(route_packet);
}

table ingress_check_fencipr {
   reads {
        standard_metadata.ingress_port : exact;  // if a match, returns the corresponding LIK for the packet's ingress_port
   }
   actions {
                check_fencipr;  // action to take if a match found 'reads' above (automatically passes LIK found from match above)
                drop_pkt;           // action to take if NO match from 'reads' above
   }
   size : 512;
}
```

FENC-IPR – P4 Code Samples

- *Ingress control logic*

```
table decrypt_fencipr {
   reads {
                  fencipr_metadata.lik_in : exact;
                  fencipr_metadata.crypto_select : exact;
   }
   actions {
                  decrypt_and_populate_fencipr;  // action if a match found 'reads' above (automatically passes LIK  from match above)
                  drop_pkt;            // action to take if NO match from 'reads' above
   }
   size : 512;
}

table ingress_check_layer3 {
   reads {
      fencipr.fpid : exact;
   }
   actions {
         check_layer3;
      drop_pkt;
   }
   size : 512;
}
```

FENC-IPR – P4 Code Samples

- *Ingress control logic*

```
table decrypt_fencipr {
   reads {
                fencipr_metadata.lik_in : exact;
                fencipr_metadata.crypto_select : exact;
   }
   actions {
                decrypt_and_populate_fencipr;  // action if a match found 'reads' above (automatically passes LIK  from match above)
                drop_pkt;            // action to take if NO match from 'reads' above
   }
   size : 512;
}
```

```
table ingress_check_layer3 {
   reads {
      fencipr.fpid : exact;
   }
   actions {
         check_layer3;
      drop_pkt;
   }
   size : 512;
}
```

```
table decrypt_layer3 {
   reads {
         fencipr_metadata.crypto_select : exact;
   }
   actions {
          decrypt_and_populate_layer3;
          drop_pkt;
   }
   size : 512;
}
```

FENC-IPR – P4 Code Samples

- *Ingress control logic*

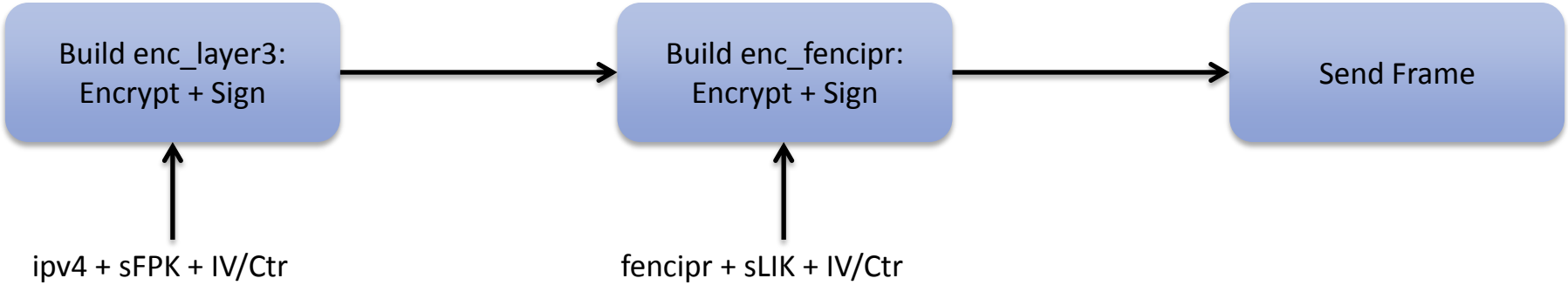
```
action check_fencipr(lik_in) {
        modify_field(fencipr_metadata.crypto_select, ETHERTYPE_FENCIPR);
        modify_field(fencipr_metadata.lik_in, lik_in);
        calc_shortsig(fencipr_metadata.calcd_shortsig, enc_fencipr.enc_bytes, ENC_FENCIPR_BYTE_WIDTH, fencipr_metadata.lik_in,
KEY_BYTE_WIDTH);
}

action decrypt_and_populate_fencipr() {
        // to get here, fencipr_metadata.crypto_select and fencipr_metadata.lik_in must be valid
        modify_field(fencipr_metadata.layer3_ethertype, fencipr.next_type);
        decrypt_aes128_cbc(fencipr, enc_fencipr, ENC_FENCIPR_BYTE_WIDTH, fencipr_metadata.lik_in, enc_fencipr.this_iv);
}

action check_layer3(fpk) {
        modify_field(fencipr_metadata.crypto_select, ETHERTYPE_IPV4);
        modify_field(fencipr_metadata.fpk, fpk);
        calc_shortsig(fencipr_metadata.calcd_shortsig, enc_layer3.enc_bytes, ENC_LAYER3_BYTE_WIDTH, fencipr_metadata.fpk,
KEY_BYTE_WIDTH);
}
```

FENC-IPR – P4 Processing Flow

- *Egress Control*



FENC-IPR – P4 Code Samples

- *Egress control logic*

```
control egress {
  apply(egress_layer3);
   apply(egress_fencipr);
   apply(send_frame);
}

table egress_layer3 {
   reads {
      // something is wrong with reading and matching these fields (it keeps missing on correct field values???)
      //   for now -> have set the default to action 'build_enc_ipv4' in 'commands.txt'
      //   change default action back to 'drop_pkt' when we figure out the problem
      fencipr_metadata.crypto_select: exact;
   }
   actions {
                  build_enc_ipv4;  // action to take if a match found 'reads' above (automatically passes LIK found from match above)
                  drop_pkt;            // action to take if NO match from 'reads' above
   }
   size : 512;
}
```

FENC-IPR – P4 Code Samples

- *Egress control logic*

```
table egress_fencipr {
   reads {
         fencipr_metadata.crypto_select : exact;
   }
   actions {
         build_enc_fencipr;  // action to take if a match found 'reads' above (automatically passes LIK found from match above)
         drop_pkt;           // action to take if NO match from 'reads' above
   }
   size : 512;
}

table send_frame {
   reads {
      routing_table.nhop_ipv4 : exact;
   }
   actions {
       set_macs;
       drop_pkt;
   }
   size : 512;
}
```

FENC-IPR – P4 Code Samples

- *Egress control logic*

```
action build_enc_ipv4(fpid) {
    modify_field(fencipr_metadata.crypto_select, ETHERTYPE_IPV4);
    encrypt_aes128_cbc(enc_layer3, ipv4, ENC_LAYER3_BYTE_WIDTH, fencipr_metadata.fpk, fencipr.aesctr);
    calc_shortsig(fencipr_metadata.calcd_shortsig, enc_layer3.enc_bytes, ENC_LAYER3_BYTE_WIDTH, fencipr_metadata.fpk,
                KEY_BYTE_WIDTH);
    modify_field(enc_fencipr.next_shortsig, fencipr_metadata.calcd_shortsig);
    // get ready for next encryption action
    modify_field(fencipr_metadata.crypto_select, 0x88b5);
}

action build_enc_fencipr(lik_out) {
    modify_field(fencipr_metadata.crypto_select, ETHERTYPE_FENCIPR);
    modify_field(fencipr_metadata.lik_out, lik_out);
    // update_iv(enc_fencipr.this_iv)  // if Dickie says we should get a new IV for each packet
    encrypt_aes128_cbc(enc_fencipr, fencipr, ENC_FENCIPR_BYTE_WIDTH, fencipr_metadata.lik_out, enc_fencipr.this_iv);
    calc_shortsig(fencipr_metadata.calcd_shortsig, enc_fencipr.enc_bytes, ENC_FENCIPR_BYTE_WIDTH, fencipr_metadata.lik_out,
                KEY_BYTE_WIDTH);
    modify_field(enc_fencipr.this_shortsig, fencipr_metadata.calcd_shortsig);
}
```

FENC-IPR – bmv2 Code Samples

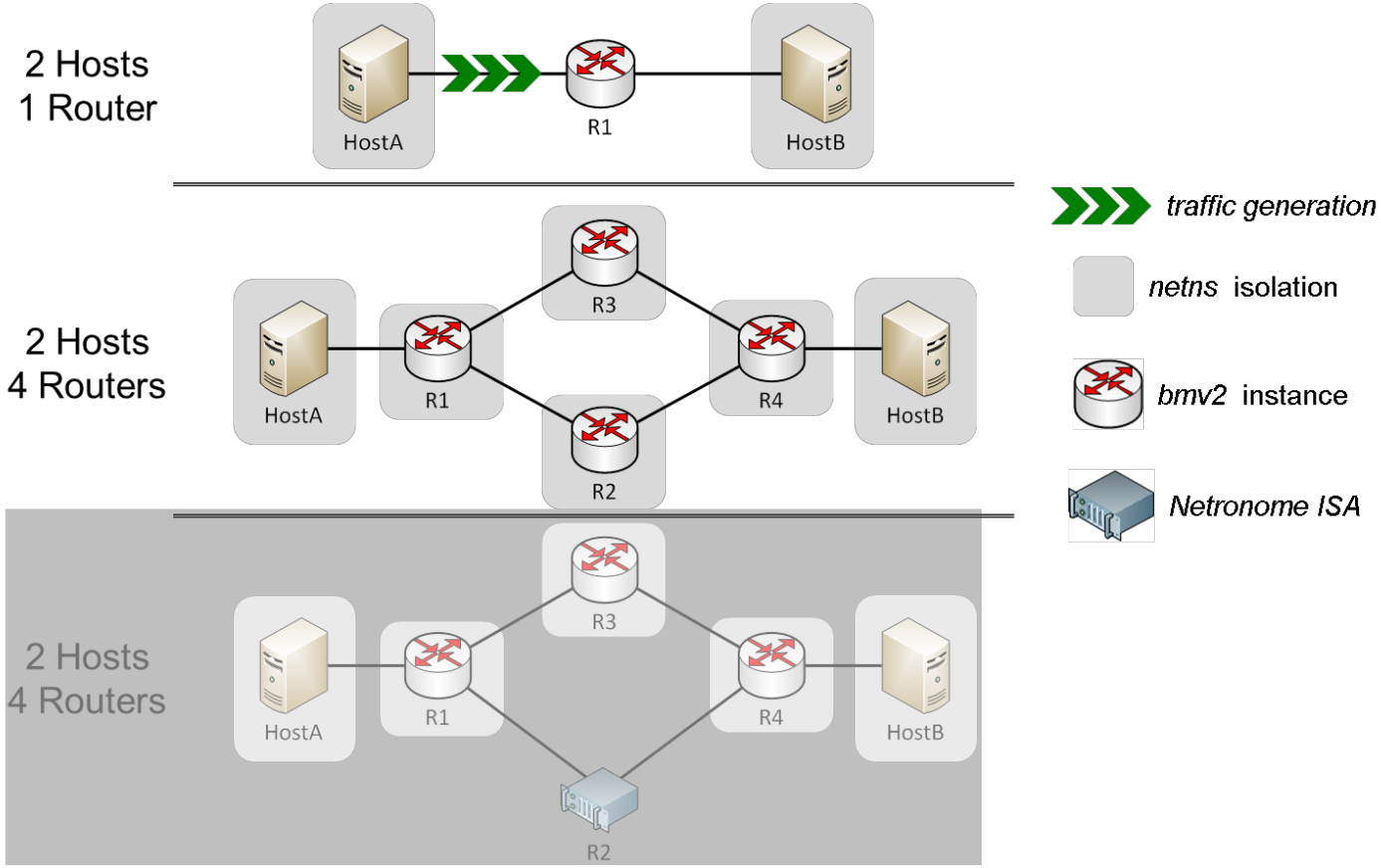
From primitives.cpp:

```
class calc_shortsig : public ActionPrimitive<Data &, const Data &, const Data &, const Data &, const Data &> {
        void operator ()(Data &dst, const Data &src, const Data &srcsize, const Data &key, const Data &keysize) {
        dst.calc_shortsig(src, srcsize, key, keysize);
 }
};
REGISTER_PRIMITIVE(calc_shortsig);
```

From data.h:

```
void calc_shortsig(const Data &src, const Data &srcsize, const Data &key, const Data &keysize) { . . . }
```

FENC-IPR Prototype – Test Networks



OpenNFP

Thank You!