Security 19: IKE

i.g.batten@bham.ac.uk

Key Management

- IPsec allows us to use keys agreed between parties, identified by SPI, to encrypt and sign packets.
- Those keys can be manually configured, or automatically set up

Manual Keying

- Usually done with endpoints and (optionally) a protocol
- “Allocate this flow an SPI, and set up the keying like this”

```
example# ipseckey
ipseckey> add ah spi 0x90125 src me.domain.com dst you.domain.com \
          authalg md5 authkey 1234567890abcdef1234567890abcdef
ipseckey> update ah spi 0x90125 dst you.domain.com hard_bytes \
          16000000
ipseckey> exit
```

Why manual?

- Simpler
- You control and generate the key material so can use your super-special RNG
- Re-keying and so on is under manual control
- Fewer protocols to assure

Why not manual?

- Involves generating and conveying key material in a secure manner
- Synchronising re-keying to avoid dropped packets might be a problem (depends on tight clock sync or some in- or out-of-band signalling)
- Hideous as number of nodes rises
- Forward secrecy tricky (although not impossible)

IKE

- Internet Key Exchange
- Based on Diffie-Hellman exchange
- Designed by people who knew what they were doing
- “Oakley” protocol presumably named after the former GCHQ site in north Cheltenham (cf. Benhall, the southern site that is now the main base)

Problems

- Key exchange has to take place *ex nihilo*, because the keys are the basis for later secure communication: we can't use IPsec (no keys)
- IPsec developed earlier than TLS, so no ability to use TLS
- Crucially, doesn't assume any sort of PKI, although it can use one if available

Problems with DH

- Man in the middle
- Assurance of Key derivation functions
- But only game in town for the purposes we have
  - (IKE supports both EC and modular variants, and plugging in other key exchange protocols would be possible)

Basic IKE flow

- Create an ISAKMP SA (*Internet Security Association Key Management Protocol Security Association*) (“*Phase One*”)
- Use the ISAKMP SA to protect the negotiation of keys used for traffic (“*Phase Two*”)
- Very low volumes of traffic over the ISAKMP SA, so ultimate quality of keys has different priorities: a break is VERY serious, but the volumes of ciphertext available to the attacker are small and the protocol protects against chosen/adaptive attacks.

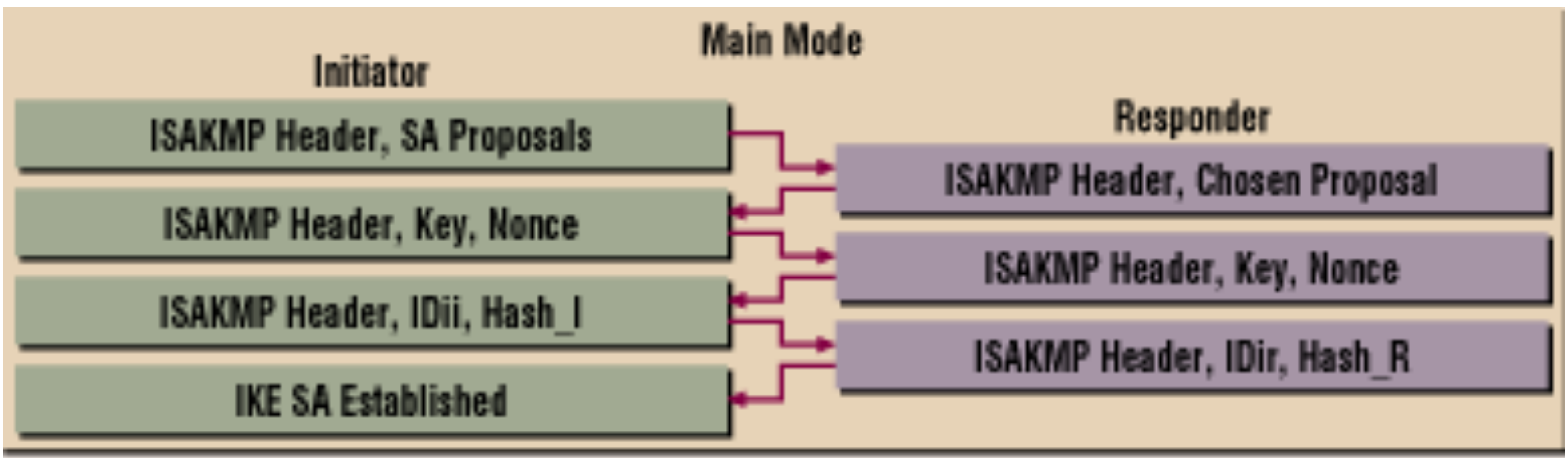
Authentication Mechanisms

- Pre-shared keys: both parties have a symmetric key that they use to authenticate while building the ISAKMP SA
- RSA Keys: one party has the private, one party the public part of an RSA key pair
- X.509 certificate: signed RSA keys

IKE Mode: Main

- Slower, involving five exchanges
- Conceals the parties' identities within the protocol

IKE Main Mode



SA Proposals

- Which algorithms are going to be used (initiator lists the ones they are willing/able to use, responder sends back the chosen algorithms)
- Nota Bene: these algorithms are protecting the ISAKMP SA, not the eventual flow of IPsec traffic, so slow is OK
    - May be different to final data algorithms because we are in user-space, not in hardware-assist or kernel space

SA Key Material

- Key material in second exchange is the two sides of a Diffie Hellman exchange, from which a shared key is derived to use to protect the rest of the exchange using the agreed symmetric algorithm

Authentication

- Both sides then send their identity, and a hash over the whole message calculated using the key material the two parties share (pre-shared key or nonces encoded using RSA)
- Both parties now share a key and are confident about the identity of the other party.
- The hash input also includes the shared key, so any man-in-the middle is detected at this point

Simple MITM

- Alice sends g^a, Bob sends g^b, both can compute g^(ab)
- Mallory can do this exchange separately with Alice and with Bob, so each negotiates a different key. Mallory then decodes traffic from Alice and re-encodes it with the key negotiated with Bob.

MITM protection

- Alice and Bob share secret S.
- Alice computes K = g^(ab) and then computes h(K . S) and sends it to Bob
- Bob does the same thing
- Mallory cannot forge the required packets as Mallory does not know S.
- S only needs to resist attack on hash function, rather than being exposed as a long-term key.

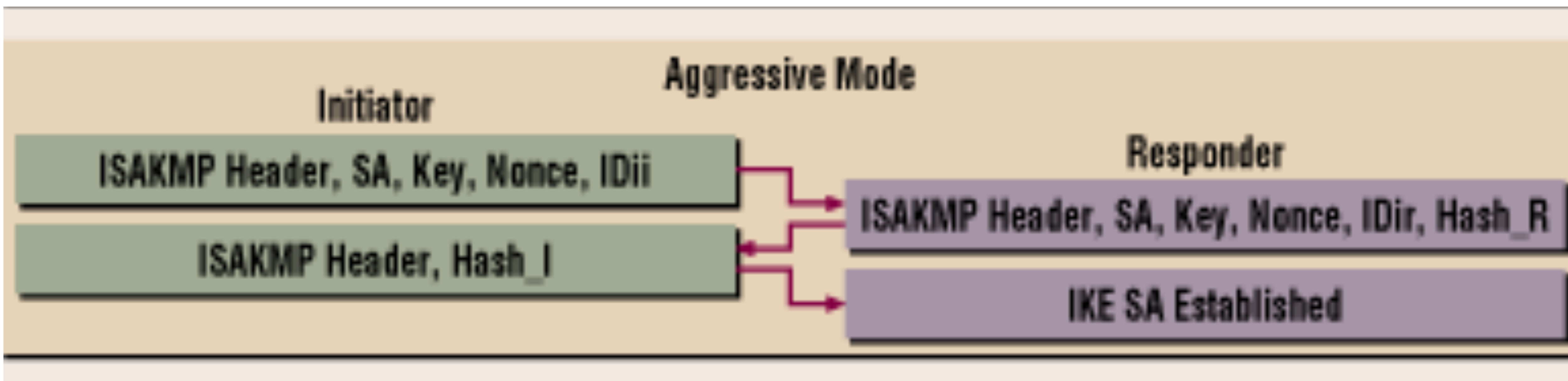
IKE Authentication

- Pre-shared keys easy
- RSA versions I suggest you read the specifications
- X.509 good because it allows authentication based on signatures on certificate, rather than IKE agent needing full details of all peers

IKE Aggressive Mode

- Main mode is slow, but protects the identities of the parties.
- This might not be a worthwhile trade-off
- Alternative is aggressive mode

IKE Aggressive Mode



Aggressive

- Packs much of the same data into three packets
- Exposes both parties’ identities, as those are sent pre-key agreement (may not matter in real-world environments)
- Doesn’t provide for delicate negotiation of algorithms (may not matter amongst friends).

Phase 2: Main Mode

- Main means to establish IPsec keys
- If “Perfect Forward Secrecy” is enabled, new DH key is established each time a new IPsec key is derived (little reason to not use this)
- Three packet exchange of DH contributions, then a key confirmation packet

Configuration Example (Racoon/KAME/ipsec-tools)

Phase 2

```
sainfo anonymous
{
        pfs_group 2;
        lifetime time 24 hour ;
        encryption_algorithm aes 192, aes 128;
        authentication_algorithm hmac_sha256, hmac_sha1 ;
        compression_algorithm deflate ;
}

remote anonymous {
   exchange_mode aggressive, main, base;
   my_identifier asn1dn;
   peers_identifier asn1dn;
   verify_identifier on;
   certificate_type x509 "cert.pem" "key.pem" ;
   ca_type x509 "cacert.pem" ;
   lifetime time 14400 sec;
   nonce_size 40;
   proposal {
     dh_group 5;
     encryption_algorithm aes 256;
     hash_algorithm sha256;
     authentication_method rsasig;
   }
}
```

Phase 1

But needs flows defining separately

```
spdadd -n 95.46.198.9 81.187.150.211 any -P out ipsec esp/transport//require;
spdadd -n 81.187.150.211 95.46.198.9 any -P in ipsec esp/transport//require;
spdadd -n 95.46.198.9 81.187.150.213 any -P out ipsec esp/transport//require;
spdadd -n 81.187.150.213 95.46.198.9 any -P in ipsec esp/transport//require;
spdadd -n 95.46.198.9 81.187.150.210 any -P out ipsec esp/transport//require;
spdadd -n 81.187.150.210 95.46.198.9 any -P in ipsec esp/transport//require;
spdadd -n 95.46.198.9 81.187.150.215 any -P out ipsec esp/transport//require;
spdadd -n 81.187.150.215 95.46.198.9 any -P in ipsec esp/transport//require;
spdadd -n 95.46.198.9 81.187.150.217 any -P out ipsec esp/transport//require;
spdadd -n 81.187.150.217 95.46.198.9 any -P in ipsec esp/transport//require;
```

Or Solaris

```
# 3 days
ikesa_lifetime_secs 259200
# one hour
childsa_lifetime_secs 3600
# 51 minutes (1h - strongswan's 9m default margin)
childsa_softlife_secs 3060

{
  label "everything we do (v4)"
  auth_method cert
  remote_id ANY
  required_issuer DN="C=GB, ST=England, L=Birmingham, O=batten.eu.org private CA, OU=VPN"
  local_id DN="C=GB, ST=England, O=batten.eu.org, OU=VPN, CN=mail.batten.eu.org"
  local_addr 147.188.192.250
  remote_addr 0.0.0.0/0
  ikesa_xform { dh_group 15 auth_alg sha256 encr_alg aes }
  childsa_pfs 5
}
```

But needs separate SA configuration (including algorithms)

```
{ laddr 147.188.192.250 raddr 81.187.150.211 } ipsec {encr_algs aes-gcm(128..256) sa
shared ike_version 2}
{ laddr 147.188.192.250 raddr 81.187.150.213 } ipsec {encr_algs aes-gcm(128..256) sa
shared ike_version 2}
{ laddr 147.188.192.250 raddr 81.187.150.210 } ipsec {encr_algs aes-gcm(128..256) sa
shared ike_version 2}
{ laddr 147.188.192.250 raddr 81.187.150.215 } ipsec {encr_algs aes-gcm(128..256) sa
shared ike_version 2}
{ laddr 147.188.192.250 raddr 81.187.150.217 } ipsec {encr_algs aes-gcm(128..256) sa
shared ike_version 2}
```

Or StrongSwan

```
conn %default
        ikelifetime=72h
        keylife=1h
        rekeymargin=9m
        keyingtries=%forever
        keyexchange=ikev2
        # group 15 for IKE
        ike=aes256-sha256-modp3072
        # group 5 for PFS
        esp=aes128-sha1-modp1536
        leftcert=cert.pem
        type=transport
        auto=start
        rightid="C=GB, ST=England, O=batten.eu.org, OU=VPN, CN=*"
        dpdaction=hold
        dpddelay=0
        mobike=no


conn mailv4
        left=95.46.198.9
        right=147.188.192.250
```

Problems with IKE

- Very poor standardisation of implementations
  - Tunnel mode, in particularly, difficult to interwork
- Configuration files are cryptic beyond belief
- Implementations on routers are again different

Problems with IKE

- Relies on DH key exchanges generating “good” keys
  - Unknown quantity, and notably some classified networks use manual keying
- Distribution of pre-shared keys difficult
- Generation and signing of certificates has usual problems (using CRLs is particularly fraught)

IKE Redux: IKEv2

- IKEv2 now becoming available, but interworking again hellish.
- Proposed 2005 (RFC 4306) but slow to get traction.
- Arrives in OSX, iOS in late 2015, used in some standalone clients before then.
- Interworking is hellish, let me restate.

IKEv2

- Removes main/aggressive distinction, and replaces it with a four-packet exchange which authenticates IKE instances **and** agrees first keys for first SA (now CHILD_SA).
- Standardises a lot of established but non-documented behaviour, including XAUTH and other features useful for VPNs.
- Much easier to use for a VPN directly, rather than tunnelling L2TP (or whatever) through it.
- But…

IKEv2

- Configuration complex, and for phones often has to be done via configuration applications rather than GUIs: problematic for BYOD.
- Fewer options but still tricky to get working, and the support for VPNs gives more opportunity for incompatibility.
- Needed for mobility, amongst other things
- You can mix IKEv2 and IKEv1 on the same platform, although it can be messy

IPsec attacks

- Not used widely outside classified networks
- Using IPsec to lift 334 network to 554 still has IL3 protection from even seeing the traffic
  - Therefore not widely attacked

IPsec Attacks

- Basic IPsec architecture simple, robust and uses proven block ciphers
- IKE uses multiple public key mechanisms, complex key exchange, lots of difficult stuff
- My suspicion is that if you were going to attack IPsec, you attack IKE first

IPsec Woes

- Poor standardisation of SHA2 hash truncation, so you can agree cipher suites but it doesn't work
  - Should truncate SHA2 to half length (256->128, 512->256) but tendency to truncate everything to 160 bits , or maybe 96, or something
  - Linux <= 2.6.32 has this problem, and is still in use, so other kit perpetuates bug for compatibility
- Similar problems for GCM.
- End up using AES+SHA1 for everything
- GCM faster if you can get it working