RFC5201-bis
Crypto Agility for HIP

Tobias Heer
Distributed Systems Group,
RWTH Aachen University

Robert Moskowitz
ICSAlabs, an Independent Div of
Verizon Business Systems
Crypto Agility

• No hard-coded ciphers/algorithms
• Negotiation of algorithms
• Possibility of extending set of used algorithms

• Goal of the changes in 5201-bis:
  – Keep changes minimal
Cryptographic Algorithms in HIP

1. HIT & HI
   a) Responder
   b) Initiator
2. Diffie Hellman (DH)
3. MAC / HMAC for control packets
4. Puzzle
5. Symmetric key derivation (keymat)
1.a Crypto Agility for the HIT

- HIT is exposed to the application
- Currently good legacy application support
  - Minimal changes to make applications HIP aware
  - Good support for DNS and referrals

→ We would like to keep it this way
Crypto Agility for the HIT (cont’d)

• Problem: Application binds to HIT before BEX
  – HIT algorithms cannot be negotiated during BEX
  – Invalid combinations of source and destination HIT are possible

• Solution: Only provide valid choices to application
  – Resolver filters results
  – All HITs the application sees are supported and usable

• Requirement: a-priori knowledge about HIT generation algorithms
  – Signature algorithm, hash function, truncation
Options for Transmitting HIT Alg. Information

• Pull from DNS
  ✗ HIT can stay as it is
  ✖ Referrals break – applications that pass on IPv6 addresses must be modified
    - HIT cannot be used alone

• Encode in HIT
  ✗ Everything stays as it is
  ✖ Slightly reduced security – only few bits to use
    - Proposal: sacrifice four bits
  ➔ Decision on ML to use this option
HIT Suites

• HIT is a product of
  – Family of signature algorithms (including groups, key lengths, ...)
  – Hash function / compression function
  – Truncation

• Goal: Avoid picking the wrong destination HIT

• HIT suite defines a selection of these algorithms
• Suite ID encodes selection (in HIT)
How to Assign Suite IDs

- 16 values only
- Encode all suites that will ever be used
- Creating new suites
  - Be selective (do not use product of all possibilities)
  - Groups of algorithms if feasible
    ("RSA+DSA+SHA-256" vs. "RSA+SHA-256" and "DSA+SHA-256")
- Deprecate and reuse suite IDs
  - Goal: long time between deprecation and reuse
  - Worst case: more than 16 suites needed
    \(\Rightarrow\) use more bits (c.f. Appendix A,B)
1.b Initiator HIT Selection

• Responder must implement Initiator’s HIT suite
• Two options:
  a) Initiator HIT = Responder HIT
     – Responder changes HIT – Initiator must change, too.
     – Breaks ACLs, certificates, etc.
  b) Negotiate during BEX
     – Must avoid downgrade attacks
     – Negotiation in signed part of the packet

• a) is problematic → we chose b)
HIT Suite List

• Responder sends list of supported HIT suites in signed part of R1
• Initiator can restart if currently chosen HIT is incompatible with HIT suite list
• List entries reflect HIT suite encoding
  – Four or more bits
  – Length field for bit-alignment: probably not
  – 8-bit HIT suite IDs in the list: 4 bits in reserve
Detect & Restart

• Downgrade attack: attacker forces Initiator and Responder to use weaker cipher than necessary
• Way of handling downgrade attacks: detect and restart BEX.
  – HIT suite list is signed, restart BEX if needed
• Replay attack: reuse old R1 with weak ciphers
  – Puzzle will not match
  – DH shared key will not match
  → No successful HIP association
• Performance of D&R: bad but it doesn’t matter
2. Crypto Agility for DH

- Currently **two** DH public keys in DH parameter
- Do we need more flexibility?
  - More DH algorithms (ECDH, non-DH) at some point?
- If yes:
  - SIGMA compliance should be maintained
  - Start negotiation in I1 (DH algorithm list)
  - R1 pre-creation is tricky:
    - hashes or hash trees as solution (c.f. Appendix C)
  - Avoid downgrade attacks
3/4. Other Algorithms

- HMAC hash function for the control channel
  - Determined by Responder’s HIT suite
- RHASH (puzzle, solution, keymat)
  - Determined by Responder’s HIT suite
- Problem: future suites without hash function
  - Other message authentication code?
  - Puzzle replacement?
  - Different key material generation?
- Solution: Remove RHASH references
  - Reference HIT suite directly
  - No single algorithm for puzzle and keymat
  - Make main text of 5201-bis algorithm agnostic
5. Key Material

• Update key material generation to draft-krawczyk-hkdf-01.txt

• Problem: salt should be as long as output

• Current salt: puzzle I and J (each 64 bits)
  – I is random and determined by Responder
  – J is a solution to I but Initiator can influence J

• Proposed solution increase size for I and J to size of hash function used in key material generation.
Conclusion

• Crypto agility for HIT & HI, DH, MAC / HMAC, puzzle, and symmetric key derivation
• Aim: keep changes small
• Enable future evolution of HIP
• Milestone for mentioned changes:  
  – IETF 78 Maastricht
Appendix

• This is a set of additional slides that illustrate some of the mentioned problems in greater detail.
Appendix A: HIT Suite Lifecycle

• How should the four bits in the ORCHID be reused?
  – Option a) 0000 has been deprecated long before rollover: reuse 0000
  – Option b) 0000 is still in use (worst case assumption): Use 1111 to indicate the use of more bits (e.g. 4+3+91 bits)
Appendix B:
Why we never want to use option b)

• Supporting too many HIT suites at the same time will lead to problems: Remember Babel

• Two alternatives:
  – Accept namespace segmentation
    • Transitivity problems in connectivity
    • Workarounds in other places
  – Implementation and maintenance pain
    • Everyone needs to support everything
    • Large code size, bad portability

➔ Use only few suites
Appendix C.1: R1 Pre-creation for Many DH Keys

- Problem: space in R1 is limited
  - DH keys can be long
  - High number of algorithms (no need for limit)
  - \( \rightarrow \) Many (long) DH public keys

- Trivial solution: pre-create a R1 for each key
  - Many R1s to pre-create (even if you do it smartly)
  - Problematic for Responders with many diverse Initiators
  - \( \rightarrow \) Not scalable
Appendix C.2: Candidate Solution

- Only include hashed DH key representations in signed part of R1
  1. Hashes of DH Keys: 128/256/... bits per DH key
  2. Create Merkle/hash tree over all DH keys and include root in signed part of R1, provide hash path to root in unsigned part

- Transmit actual DH key in unsigned part of R1
  - 0-ed spaceholder does not work (different key lengths)