
ike Documentation

Release 0.1.1

Kimmo Parviainen-Jalanko

Mar 12, 2018

Contents

1	About ike	3
1.1	Status	3
1.2	Design principles	3
1.3	Documentation	4
1.4	License	4
1.5	References	4
2	ike package	5
2.1	Subpackages	5
2.2	Submodules	6
2.3	ike.const module	6
2.4	ike.initiator module	9
2.5	ike.payloads module	9
2.6	ike.proposal module	11
2.7	ike.protocol module	12
2.8	Module contents	13
3	ike	15
4	Indices and tables	17
	Python Module Index	19

Contents:

The goal of this project is to be a minimalistic IKEv2 (RFC 5996) implementation in Python.

1.1 Status

This project is in early stages. Use at own risk.

It will make your IP stack talk ESP to the remote peer.

What it can do:

- Act as an initiator
- Authenticate itself and peer using raw RSA keys.
- Install ESP SAs and SPD entries to use the key material via `setkey` command from `ipsec-tools`.

Limitations (hardcoded values):

- Cipher algorithm is Camellia in CBC mode with 256 bit keys.
- HMAC / Hash / PRF algorithm is SHA2/256.
- IKE group is Diffie-Hellman modp 14.
- Authentication (both own private and peer public) key file paths are hardcoded.
- 'setkey' syntax is of whatever the ipsec-tools on Debian 7.1 accept.
- Traffic selectors are `myip:any:0-65535 <-> peerip:any:0-65535`

1.2 Design principles

- Minimal amount of code.
- Support *MUST* features of draft-kivinen-ipsecme-ikev2-rfc5996bis-02 (RFC 5996 successor)

- Use strongest algorithms possible.

1.3 Documentation

You can read the Documentation at <https://pythonhosted.org/ike>

1.3.1 What this project is *NOT* going to be

- ISAKMP (IKEv1) RFC 2409 compliant
- IPSec data plane / ESP protocol

1.4 License

- MIT License

1.5 References

- <http://tools.ietf.org/html/draft-kivinen-ipsecme-ikev2-rfc5996bis-02>
- <http://tools.ietf.org/html/draft-kivinen-ipsecme-ikev2-minimal-01>

2.1 Subpackages

2.1.1 ike.util package

Submodules

ike.util.cipher module

class ike.util.cipher.**AES** (*key*, *iv=None*)

Bases: ike.util.cipher._Cipher

algorithm

alias of *AES*

class ike.util.cipher.**Camellia** (*key*, *iv=None*)

Bases: ike.util.cipher._Cipher

algorithm

alias of *Camellia*

ike.util.cipher.**pad** (*data*, *blocksize=16*)

Pads data to blocksize according to RFC 4303. Pad length field is included in output.

ike.util.conv module

ike.util.conv.**to_bytes** (*x*)

ike.util.dh module

class ike.util.dh.**DiffieHellman** (*group=14*, *n=64*)

Bases: object

```
derivate (other_key)  
generate_private_key (n)  
generate_public_key ()  
generator = 2  
shared_secret
```

ike.util.dump module

```
ike.util.dump.dump (src)  
    Returns data in hex format in groups of 4 octets delimited by spaces for debugging purposes.
```

ike.util.external module

```
ike.util.external.run_setkey (input)  
    Runs a script through the 'setkey' command that is a user space interface for PFKEY. :param input: setkey  
    configuration file contents.
```

ike.util.prf module

```
ike.util.prf.prf (key, data, hash_algorithm='sha256')  
ike.util.prf.prfplus (key, data, n)
```

ike.util.pubkey module

```
ike.util.pubkey.sign (data, filename, hash_alg='SHA-256')  
ike.util.pubkey.verify (data, signature, filename)
```

Module contents

2.2 Submodules

2.3 ike.const module

```
class ike.const.AuthenticationType  
    Bases: enum.IntEnum  
    An enumeration.  
  
    DSS = 3  
    PSK = 2  
    RSA = 1  
  
class ike.const.ExchangeType  
    Bases: enum.IntEnum  
    An enumeration.
```

```
CREATE_CHILD_SA = 36
IKE_AUTH = 35
IKE_SA_INIT = 34
INFORMATIONAL = 37

class ike.const.MessageType
    Bases: enum.IntEnum

    An enumeration.

    ADDITIONAL_IP4_ADDRESS = 16397
    ADDITIONAL_IP6_ADDRESS = 16398
    ADDITIONAL_TS_POSSIBLE = 16386
    ANOTHER_AUTH_FOLLOWS = 16405
    AUTHENTICATION_FAILED = 24
    AUTHORIZATION_FAILED = 46
    AUTH_LIFETIME = 16403
    CHILDLESS_IKEV2_SUPPORTED = 16418
    CHILD_SA_NOT_FOUND = 44
    COOKIE = 16390
    COOKIE2 = 16401
    EAP_ONLY_AUTHENTICATION = 16417
    ERX_SUPPORTED = 16427
    ESP_TFC_PADDING_NOT_SUPPORTED = 16394
    FAILED_CP_REQUIRED = 37
    HTTP_CERT_LOOKUP_SUPPORTED = 16392
    IFOM_CAPABILITY = 16428
    IKEV2_MESSAGE_ID_SYNC = 16422
    IKEV2_MESSAGE_ID_SYNC_SUPPORTED = 16420
    INITIAL_CONTACT = 16384
    INTERNAL_ADDRESS_FAILURE = 36
    INVALID_GROUP_ID = 45
    INVALID_IKE_SPI = 4
    INVALID_KEY_PAYLOAD = 17
    INVALID_MAJOR_VERSION = 5
    INVALID_MESSAGE_ID = 9
    INVALID_SELECTORS = 39
    INVALID_SPI = 11
    INVALID_SYNTAX = 7
```

IPCOMP_SUPPORTED = 16387
IPSEC_REPLAY_COUNTER_SYNC = 16423
IPSEC_REPLAY_COUNTER_SYNC_SUPPORTED = 16421
LINK_ID = 16414
MOBIKE_SUPPORTED = 16396
MULTIPLE_AUTH_SUPPORTED = 16404
NAT_DETECTION_DESTINATION_IP = 16389
NAT_DETECTION_SOURCE_IP = 16388
NON_FIRST_FRAGMENTS_ALSO = 16395
NO_ADDITIONAL_ADDRESSES = 16399
NO_ADDITIONAL_SAS = 35
NO_NATS_ALLOWED = 16402
NO_PROPOSAL_CHOSEN = 14
PSK_CONFIRM = 16426
PSK_PERSIST = 16425
QUICK_CRASH_DETECTION = 16419
REDIRECT = 16407
REDIRECTED_FROM = 16408
REDIRECT_SUPPORTED = 16406
REKEY_SA = 16393
ROHC_SUPPORTED = 16416
Reserved = 0
SECURE_PASSWORD_METHODS = 16424
SENDER_REQUEST_ID = 16429
SET_WINDOW_SIZE = 16385
SINGLE_PAIR_REQUIRED = 34
TEMPORARY_FAILURE = 43
TICKET_ACK = 16411
TICKET_LT_OPAQUE = 16409
TICKET_NACK = 16412
TICKET_OPAQUE = 16413
TICKET_REQUEST = 16410
TS_UNACCEPTABLE = 38
UNACCEPTABLE_ADDRESSES = 40
UNEXPECTED_NAT_DETECTED = 41
UNSUPPORTED_CRITICAL_PAYLOAD = 1

```

UPDATE_SA_ADDRESSES = 16400
USE_ASSIGNED_HoA = 42
USE_TRANSPORT_MODE = 16391
USE_WESP_MODE = 16415
class ike.const.ProtocolID
    Bases: enum.IntEnum
    An enumeration.
    AH = 2
    ESP = 3
    IKE = 1

```

2.4 ike.initiator module

IKE v2 (RFC 5996) initiator implementation

Usage: initiator.py <remote_peer>

To clean up afterwards,

```
setkey -FP && setkey -F
```

```

class ike.initiator.IKEInitiator
    Bases: asyncio.protocols.DatagramProtocol
    Implements an IKE initiator that attempt to negotiate a single child SA to remote peer.
    connectionRefused()
    connection_made(transport)
    datagram_received(data, address)
ike.initiator.main(peer)

```

2.5 ike.payloads module

IKEv2 Payloads as specified in RFC 5996 sections 3.2 - 3.16

```

class ike.payloads.AUTH(signed_octets=None, data=None, next_payload=<no_next_payload: 0>,
                        critical=False)
    Bases: ike.payloads._IkePayload
    Authentication Payload
class ike.payloads.Fragment(data=None, next_payload=<no_next_payload: 0>, critical=False,
                            fragment=None)
    Bases: ike.payloads._IkePayload
    Fragment Payload
    parse(data)

```

```
class ike.payloads.IDI (data=None, next_payload=<no_next_payload: 0>, critical=False)
    Bases: ike.payloads._IkePayload

    Identification Payload for initiator

class ike.payloads.IDr (data=None, next_payload=<no_next_payload: 0>, critical=False)
    Bases: ike.payloads._IkePayload

    Identification Payload for responder

class ike.payloads.KE (data=None, next_payload=<no_next_payload: 0>, critical=False, group=14,
                        diffie_hellman=None)
    Bases: ike.payloads._IkePayload

    Key Exchange Payload

    parse (data)

class ike.payloads.Nonce (data=None, next_payload=<no_next_payload: 0>, critical=False,
                           nonce=None)
    Bases: ike.payloads._IkePayload

    Nonce Payload

    parse (data)

class ike.payloads.Notify (notify_type=None, data=None, next_payload=<no_next_payload: 0>,
                           critical=False)
    Bases: ike.payloads._IkePayload

    Notify Payload

    parse (data)

class ike.payloads.SA (data=None, proposals=None, next_payload=<no_next_payload: 0>, critical=False)
    Bases: ike.payloads._IkePayload

    Security Association Payload

    parse (data)

class ike.payloads.SK (data=None, next_payload=<no_next_payload: 0>, critical=False, iv=None,
                        ciphertext=None)
    Bases: ike.payloads._IkePayload

    Encrypted Payload

    mac (hmac)

class ike.payloads.TSi (addr=None, data=None, next_payload=<no_next_payload: 0>, critical=False)
    Bases: ike.payloads._TS

    Traffic Selector Payload for initiator

class ike.payloads.TSr (addr=None, data=None, next_payload=<no_next_payload: 0>, critical=False)
    Bases: ike.payloads._TS

    Traffic Selector Payload for responder

class ike.payloads.Type
    Bases: enum.IntEnum

    Payload types from IANA

    AUTH = 39
```

```

CERT = 37
CERTREQ = 38
CP = 47
Delete = 42
EAP = 48
Fragment = 132
GSA = 51
GSPM = 49
IDg = 50
IDi = 35
IDr = 36
KD = 52
KE = 34
Ni = 40
Nonce = 40
Notify = 41
Nr = 40
SA = 33
SK = 46
TSi = 44
TSr = 45
Vendor = 43
no_next_payload = 0

class ike.payloads.Vendor (data=None, next_payload=<no_next_payload: 0>, critical=False, vendor=None)
    Bases: ike.payloads._IkePayload
    Nonce Payload
    parse (data)

ike.payloads.get_by_type (payload_type)
    Returns an IkePayload (sub)class based on the RFC5996 payload_type :param payload_type: int() Ike Payload type

```

2.6 ike.proposal module

Implements Proposal and Transform substructures for Security association (SA) payloads.

Conforms to [RFC5996 section 3.3](#)

```

class ike.proposal.Proposal (data=None, num=1, protocol=<ProtocolID.IKE: 1>, spi=None, spi_len=0, last=False, transforms=None)
    Bases: object

```

data

parse (*data*)

class ike.proposal.**Transform** (*name, keysize=None, last=False*)

Bases: object

data

2.7 ike.protocol module

High level interface to [IKEv2 protocol](#)

class ike.protocol.**IKE** (*address, peer, dh_group=14, nonce_len=32*)

Bases: object

A single IKE negotiation / SA.

Currently implements only Initiator side of the negotiation.

auth_rcv ()

Handle peer's IKE_AUTH response.

auth_send ()

Generates the second (IKE_AUTH) packet for Initiator

Returns bytes() containing a valid IKE_INIT packet

authenticate_peer (*auth_data, peer_id, message*)

Verifies the peers authentication.

decrypt (*data*)

Decrypts an encrypted (SK, 46) IKE payload using self.SK_er

Parameters **data** – Encrypted IKE payload including headers (payloads.SK())

Returns next_payload, data_containing_payloads

Raises [IkeError](#) – If packet is corrupted.

encrypt_and_hmac (*packet*)

Encrypts and signs a Packet() using self.SK_ei and self.SK_ai

Parameters **packet** – Unencrypted Packet() with one or more payloads.

Returns Encrypted and signed Packet() with a single payloads.SK

init_rcv ()

Parses the IKE_INIT response packet received from Responder.

Assigns the correct values of rSPI and Nr Calculates Diffie-Hellman exchange and assigns all keys to self.

init_send ()

Generates the first (IKE_INIT) packet for Initiator

Returns bytes() containing a valid IKE_INIT packet

install_ipsec_sas ()

parse_packet (*data*)

Parses a received packet in to Packet() with corresponding payloads. Will decrypt encrypted packets when needed.

Parameters **data** – bytes() IKE packet from wire.

Returns Packet() instance

Raises *IkeError* – on malformed packet

verify_hmac (*data*)

Verifies the HMAC signature of an encrypted (SK, 46) payload using self.SK_ar

Parameters *data* – bytes(payloads.SK())

Raises *IkeError* – if calculated signature does not match the one in the payload

exception *ike.protocol.IkeError*

Bases: *Exception*

class *ike.protocol.Packet* (*data=None, exchange_type=None, message_id=0, iSPI=0, rSPI=0*)

Bases: *object*

An IKE packet.

To generate packets:

1. instantiate an Packet()
2. add payloads by Packet.add_payload(<payloads.IkePayload instance>)
3. send bytes(Packet) to other peer.

Received packets should be generated by IKE.parse_packet().

add_payload (*payload*)

Adds a payload to packet, updating last payload's next_payload field

class *ike.protocol.State*

Bases: *enum.IntEnum*

An enumeration.

AUTH = 2

INIT = 1

STARTING = 0

2.8 Module contents

CHAPTER 3

ike

CHAPTER 4

Indices and tables

- `genindex`
- `modindex`
- `search`

i

- ike, 15
 - ike.const, 6
 - ike.initiator, 9
 - ike.payloads, 9
 - ike.proposal, 11
 - ike.protocol, 12
 - ike.util, 6
 - ike.util.cipher, 5
 - ike.util.conv, 5
 - ike.util.dh, 5
 - ike.util.dump, 6
 - ike.util.external, 6
 - ike.util.prf, 6
 - ike.util.pubkey, 6

A

add_payload() (ike.protocol.Packet method), 13
 ADDITIONAL_IP4_ADDRESS (ike.const.MessageType attribute), 7
 ADDITIONAL_IP6_ADDRESS (ike.const.MessageType attribute), 7
 ADDITIONAL_TS_POSSIBLE (ike.const.MessageType attribute), 7
 AES (class in ike.util.cipher), 5
 AH (ike.const.ProtocolID attribute), 9
 algorithm (ike.util.cipher.AES attribute), 5
 algorithm (ike.util.cipher.Camellia attribute), 5
 ANOTHER_AUTH_FOLLOWS (ike.const.MessageType attribute), 7
 AUTH (class in ike.payloads), 9
 AUTH (ike.payloads.Type attribute), 10
 AUTH (ike.protocol.State attribute), 13
 AUTH_LIFETIME (ike.const.MessageType attribute), 7
 auth_recv() (ike.protocol.IKE method), 12
 auth_send() (ike.protocol.IKE method), 12
 authenticate_peer() (ike.protocol.IKE method), 12
 AUTHENTICATION_FAILED (ike.const.MessageType attribute), 7
 AuthenticationType (class in ike.const), 6
 AUTHORIZATION_FAILED (ike.const.MessageType attribute), 7

C

Camellia (class in ike.util.cipher), 5
 CERT (ike.payloads.Type attribute), 10
 CERTREQ (ike.payloads.Type attribute), 11
 CHILD_SA_NOT_FOUND (ike.const.MessageType attribute), 7
 CHILDLESS_IKEV2_SUPPORTED (ike.const.MessageType attribute), 7
 connection_made() (ike.initiator.IKEInitiator method), 9
 connectionRefused() (ike.initiator.IKEInitiator method), 9
 COOKIE (ike.const.MessageType attribute), 7

COOKIE2 (ike.const.MessageType attribute), 7
 CP (ike.payloads.Type attribute), 11
 CREATE_CHILD_SA (ike.const.ExchangeType attribute), 6

D

data (ike.proposal.Proposal attribute), 11
 data (ike.proposal.Transform attribute), 12
 datagram_received() (ike.initiator.IKEInitiator method), 9
 decrypt() (ike.protocol.IKE method), 12
 Delete (ike.payloads.Type attribute), 11
 derivate() (ike.util.dh.DiffieHellman method), 5
 DiffieHellman (class in ike.util.dh), 5
 DSS (ike.const.AuthenticationType attribute), 6
 dump() (in module ike.util.dump), 6

E

EAP (ike.payloads.Type attribute), 11
 EAP_ONLY_AUTHENTICATION (ike.const.MessageType attribute), 7
 encrypt_and_hmac() (ike.protocol.IKE method), 12
 ERX_SUPPORTED (ike.const.MessageType attribute), 7
 ESP (ike.const.ProtocolID attribute), 9
 ESP_TFC_PADDING_NOT_SUPPORTED (ike.const.MessageType attribute), 7
 ExchangeType (class in ike.const), 6

F

FAILED_CP_REQUIRED (ike.const.MessageType attribute), 7
 Fragment (class in ike.payloads), 9
 Fragment (ike.payloads.Type attribute), 11

G

generate_private_key() (ike.util.dh.DiffieHellman method), 6
 generate_public_key() (ike.util.dh.DiffieHellman method), 6
 generator (ike.util.dh.DiffieHellman attribute), 6

get_by_type() (in module ike.payloads), 11
GSA (ike.payloads.Type attribute), 11
GSPM (ike.payloads.Type attribute), 11

H

HTTP_CERT_LOOKUP_SUPPORTED
(ike.const.MessageType attribute), 7

I

IDg (ike.payloads.Type attribute), 11
IDi (class in ike.payloads), 9
IDi (ike.payloads.Type attribute), 11
IDr (class in ike.payloads), 10
IDr (ike.payloads.Type attribute), 11
IFOM_CAPABILITY (ike.const.MessageType attribute), 7
IKE (class in ike.protocol), 12
IKE (ike.const.ProtocolID attribute), 9
ike (module), 13, 15
ike.const (module), 6
ike.initiator (module), 9
ike.payloads (module), 9
ike.proposal (module), 11
ike.protocol (module), 12
ike.util (module), 6
ike.util.cipher (module), 5
ike.util.conv (module), 5
ike.util.dh (module), 5
ike.util.dump (module), 6
ike.util.external (module), 6
ike.util.prf (module), 6
ike.util.pubkey (module), 6
IKE_AUTH (ike.const.ExchangeType attribute), 7
IKE_SA_INIT (ike.const.ExchangeType attribute), 7
IkeError, 13
IKEInitiator (class in ike.initiator), 9
IKEV2_MESSAGE_ID_SYNC (ike.const.MessageType attribute), 7
IKEV2_MESSAGE_ID_SYNC_SUPPORTED
(ike.const.MessageType attribute), 7
INFORMATIONAL (ike.const.ExchangeType attribute), 7
INIT (ike.protocol.State attribute), 13
init_rcv() (ike.protocol.IKE method), 12
init_send() (ike.protocol.IKE method), 12
INITIAL_CONTACT (ike.const.MessageType attribute), 7
install_ipsec_sas() (ike.protocol.IKE method), 12
INTERNAL_ADDRESS_FAILURE
(ike.const.MessageType attribute), 7
INVALID_GROUP_ID (ike.const.MessageType attribute), 7
INVALID_IKE_SPI (ike.const.MessageType attribute), 7

INVALID_KE_PAYLOAD (ike.const.MessageType attribute), 7
INVALID_MAJOR_VERSION (ike.const.MessageType attribute), 7
INVALID_MESSAGE_ID (ike.const.MessageType attribute), 7
INVALID_SELECTORS (ike.const.MessageType attribute), 7
INVALID_SPI (ike.const.MessageType attribute), 7
INVALID_SYNTAX (ike.const.MessageType attribute), 7
IPCOMP_SUPPORTED (ike.const.MessageType attribute), 7
IPSEC_REPLAY_COUNTER_SYNC
(ike.const.MessageType attribute), 8
IPSEC_REPLAY_COUNTER_SYNC_SUPPORTED
(ike.const.MessageType attribute), 8

K

KD (ike.payloads.Type attribute), 11
KE (class in ike.payloads), 10
KE (ike.payloads.Type attribute), 11

L

LINK_ID (ike.const.MessageType attribute), 8

M

mac() (ike.payloads.SK method), 10
main() (in module ike.initiator), 9
MessageType (class in ike.const), 7
MOBIKE_SUPPORTED (ike.const.MessageType attribute), 8
MULTIPLE_AUTH_SUPPORTED
(ike.const.MessageType attribute), 8

N

NAT_DETECTION_DESTINATION_IP
(ike.const.MessageType attribute), 8
NAT_DETECTION_SOURCE_IP
(ike.const.MessageType attribute), 8
Ni (ike.payloads.Type attribute), 11
NO_ADDITIONAL_ADDRESSES
(ike.const.MessageType attribute), 8
NO_ADDITIONAL_SAS (ike.const.MessageType attribute), 8
NO_NATS_ALLOWED (ike.const.MessageType attribute), 8
no_next_payload (ike.payloads.Type attribute), 11
NO_PROPOSAL_CHOSEN (ike.const.MessageType attribute), 8
NON_FIRST_FRAGMENTS_ALSO
(ike.const.MessageType attribute), 8
Nonce (class in ike.payloads), 10

Nonce (ike.payloads.Type attribute), 11
 Notify (class in ike.payloads), 10
 Notify (ike.payloads.Type attribute), 11
 Nr (ike.payloads.Type attribute), 11

P

Packet (class in ike.protocol), 13
 pad() (in module ike.util.cipher), 5
 parse() (ike.payloads.Fragment method), 9
 parse() (ike.payloads.KE method), 10
 parse() (ike.payloads.Nonce method), 10
 parse() (ike.payloads.Notify method), 10
 parse() (ike.payloads.SA method), 10
 parse() (ike.payloads.Vendor method), 11
 parse() (ike.proposal.Proposal method), 12
 parse_packet() (ike.protocol.IKE method), 12
 prf() (in module ike.util.prf), 6
 prfplus() (in module ike.util.prf), 6
 Proposal (class in ike.proposal), 11
 ProtocolID (class in ike.const), 9
 PSK (ike.const.AuthenticationType attribute), 6
 PSK_CONFIRM (ike.const.MessageType attribute), 8
 PSK_PERSIST (ike.const.MessageType attribute), 8

Q

QUICK_CRASH_DETECTION (ike.const.MessageType attribute), 8

R

REDIRECT (ike.const.MessageType attribute), 8
 REDIRECT_SUPPORTED (ike.const.MessageType attribute), 8
 REDIRECTED_FROM (ike.const.MessageType attribute), 8
 REKEY_SA (ike.const.MessageType attribute), 8
 Reserved (ike.const.MessageType attribute), 8
 ROHC_SUPPORTED (ike.const.MessageType attribute), 8
 RSA (ike.const.AuthenticationType attribute), 6
 run_setkey() (in module ike.util.external), 6

S

SA (class in ike.payloads), 10
 SA (ike.payloads.Type attribute), 11
 SECURE_PASSWORD_METHODS (ike.const.MessageType attribute), 8
 SENDER_REQUEST_ID (ike.const.MessageType attribute), 8
 SET_WINDOW_SIZE (ike.const.MessageType attribute), 8
 shared_secret (ike.util.dh.DiffieHellman attribute), 6
 sign() (in module ike.util.pubkey), 6
 SINGLE_PAIR_REQUIRED (ike.const.MessageType attribute), 8

SK (class in ike.payloads), 10
 SK (ike.payloads.Type attribute), 11
 STARTING (ike.protocol.State attribute), 13
 State (class in ike.protocol), 13

T

TEMPORARY_FAILURE (ike.const.MessageType attribute), 8
 TICKET_ACK (ike.const.MessageType attribute), 8
 TICKET_LT_OPAQUE (ike.const.MessageType attribute), 8
 TICKET_NACK (ike.const.MessageType attribute), 8
 TICKET_OPAQUE (ike.const.MessageType attribute), 8
 TICKET_REQUEST (ike.const.MessageType attribute), 8
 to_bytes() (in module ike.util.conv), 5
 Transform (class in ike.proposal), 12
 TS_UNACCEPTABLE (ike.const.MessageType attribute), 8
 TSi (class in ike.payloads), 10
 TSi (ike.payloads.Type attribute), 11
 TSr (class in ike.payloads), 10
 TSr (ike.payloads.Type attribute), 11
 Type (class in ike.payloads), 10

U

UNACCEPTABLE_ADDRESSES (ike.const.MessageType attribute), 8
 UNEXPECTED_NAT_DETECTED (ike.const.MessageType attribute), 8
 UNSUPPORTED_CRITICAL_PAYLOAD (ike.const.MessageType attribute), 8
 UPDATE_SA_ADDRESSES (ike.const.MessageType attribute), 8
 USE_ASSIGNED_HoA (ike.const.MessageType attribute), 9
 USE_TRANSPORT_MODE (ike.const.MessageType attribute), 9
 USE_WESP_MODE (ike.const.MessageType attribute), 9

V

Vendor (class in ike.payloads), 11
 Vendor (ike.payloads.Type attribute), 11
 verify() (in module ike.util.pubkey), 6
 verify_hmac() (ike.protocol.IKE method), 13