Indranet Protocol Message Format

The initial idea for Indranet was just as a replacement for the Tor network, but it quickly became obvious that it can be rethought as a generic, programmable relay network.

The specification is intended to be as open ended as possible and relays process messages without any limitations on how they can be constructed, so that developers can tailor them to their use case, and importantly, so that these different uses are not mutually exclusive.

In this text there is quite a number of specialist terms related to cryptography and binary encoding schemes, see the Glossary for explanations.

Contents

Cryptographic Algorithms

Cryptographic Hash Function

SHA256

This hash function is still considered to be secure, and in Indra it uses an AVX2 implementation on AMD64 platform, which is sufficiently fast.

Blake 2 and 3 are trendy hash functions that claim higher performance, but real world performance is maybe 12% faster and has a shorter history and less attempts to break it.

🔝

Elliptic Curve Group

secp256k1

It is trendy to use the Edwards 25519 curve, and that group has slightly better properties in it's symmetries, but this curve has not been broken after far more attempts, and is not used with a weak HD derivation algorithm, keys are only derived from fresh entropy.

Indra does not have a wallet and keys are changed periodically as the number of signatures generated exceeds a conservative number.

⤴️

EC Signatures

Schnorr

For reasons of performance and data structure uniformity, Indra uses Schnorr signatures, which also provides a uniform size, and can be reduced in length from the standard 64 bytes to 48 bytes if the keys are changed more frequently and if desired can, like ECDSA signatures used in Bitcoin embed the public key where security reasons recommend cloaking it until after the cloaked form (" address") is signed and then won't be reused.

⤴️

Cloaked Public Key

In order to prevent the correlation of packets to session public keys, Indra uses a construction of 4 byte random blinding factor, which is then concatenated at the end of the 33 byte public key, and the first 4 bytes of the SHA256 hash of this combined byte slice is appended to the blinding factor prefix.

Only the relay who has the private key matching the public key in their session database can generate this matching value, any attacker cannot.

Any Crypt inadvertently delivered to a relay other than the one intended by the client will thus leak no information about the identity of the session, thus defeating attempts to correlate the session, and thus origin of a message, from evil relays seeing such messages passing through a common relay on their path.

This is unlikely to happen anyway, as point to paint connections that pass through a relay are coming from potentially many different previous hops, and the use of a relay for different positions in a path are different sessions, as well.

⤴️

Symmetric Encryption

AES-CTR Block Cipher with Schnorr Signature MAC

AES-CTR is a fast, block cipher method, with no authentication. Rather than use a plurality of authentication methods, and because Schnorr signatures are quite efficient, encryption of messages requires the inserting of the signature prior to the position indicated by the Offset of the Crypt, or in other words, at the end of the encrypted message.

EC signatures are as strong authenticity protection as it is possible to achieve, not a lesser, weaker form like used in many HMACs. The signature is based on the hash of the message, and so the message checksum need not be independently made. Either the signature is valid on the message hash, or some part of it is corrupt and the whole message is rejected and if protocol allows it, a retransmit request would be sent to try again.

Forward Error Correction

Indra uses Reed Solomon FEC with a dynamic adjustment that aims to maintain a buffer against the need for retransmit, and starts with a conservatively high redundancy ratio that adjusts slowly downwards to prevent retransmit latency.

⤴️

Field Types

Magic

This is a 4 byte string, as 4 8-bit ASCII, usually resembling or containing the message type in human readable form.

32 bit values like this are a common bit width for formatting sentinels that indicate that a specific message format follows.

It is sufficiently long as to be unlikely to occur, let alone in the protocol specified position in the message bytes.

⤴️

Private Key

This is a 32 byte, 256 bit long random value, with the limitation of it being a member of the elliptic curve group secp256k1.

⤴️

Public Key

The public key is a standard 257 bit, 33 byte public key, the additional bit being the sign of coordinate of the key.

Signature

A standard 64 bit Schnorr signature.

⤴️

Cloaked Public Key

As described elsewhere, the Cloaked Public Key is constructed using a 4 byte random blinding factor that is appended to the public key bytes, hashed, and the first 4 bytes of this hash are appended to the blinding factor.

⤴️

Integers

All standard integer types are supported, and are encoded with Little Endian byte ording that is native to most modern CPUs.

In addition to the 8, 16, 32 and 64 bit values, which can also be signed, there is a 3 byte long, 24 bit value used in several messages for a maximum length of 16 Megabytes, being a reasonable maximum message payload size.

⤴️

Initialisation Vector

16 bytes long Initialisation Vectors, the most common standard AES encryption Initialisation Vectors are used in Indra for symmetric encryption using ECDH derived public/private keys.

⤴️

Symmetric Key

A symmetric key is the secret, 32 bytes long, 256 bits, that is used with an Initialisation Vector and a cipher. In Indra, as mentioned in the Symmetric Encryption glossary entry, this is used with AES-CTR block/stream cipher mode, and secured with a Schnorr signature.

ID

ID is a random 64 bit, 8 byte long field that is used to identify a request so that it can be quickly retrieved from the pending responses, and anywhere that a signature needs to be made on data, to ensure the hash that is signed on is not repeated.

Timestamp

Timestamps are 64 bit, unsigned integers that are interpreted as 64 bit Unix timestamps, the number of seconds since the first second of the year 1970.

MilliSatoshi

64 bit value representing 1/1000th of a Satoshi, being 1/100000000th (hundred millionth) of a bitcoin.

Message types

Session

Byte length	Name	Description
4	Magic	`sess`
32	Header	Symmetric ECDH private key used in reply headers and simple forwards
32	Payload	Symmetric ECDH private key used for encrypting response payloads

The session is the most important and primary message type in the sense that it must be delivered in order for a relay to be obliged to perform services for clients.

Sessions contain a Header key and a Payload key, which is described in the Reply section in the following.

The session is a reference to a pre-paid balance, against which a bytes/time rate is applied to messages that are forwarded via the session.

The session also can be alternatively billed on a different rate for Exit messages, as described elsewhere.

⤴️

Forward

Byte length	Name	Description
4	Magic	`frwd`
33	Relay	Public identity key of relay to forward this message to

The number one task of Indranet relays is to accept a message, and forward it to another relay. They do this only under the proviso that there is a session that has been established and paid for using the Session.

The Indra protocol is connectionless because relays do not participate in making routing decisions, and thus also the low level network transport used does not have a handshake for the base case of relaying a message according to instructions in the decrypted part at the head of the message.

⤴️

Crypt

Byte length	Name	Description
4	Magic	`cryp`
8	Cloaked* Header public key	Indicates to the valid receiver which public key is being used.
33	Message public key	(sender generated one-time)
16	Initialisation Vector	standard AES high entropy random value)
3	Offset	24 bit vector (up to 16Mb header) for beginning of payload (using Payload key from Session)

The second most important message type in Indranet is the Crypt.

The crypt is an encrypted message, consisting of a header containing a cloaked session key referring to the session private key, and the random seed value used to prevent the possibility of plaintext cryptanalysis attacks.

The crypt specifies encryption that is used to "wrap" the remainder of a message so that only the intended recipient can see it, a combination of encryption and authentication rolled into one.

The `Offset` field is encrypted as the first 3 bytes of the encryption that uses the Header key, indicates the point at which the use of this key ends and the second key, the Payload key is used for the remainder. If this value is zero, there is no boundary and the Header key is to be used up to the end of the message. This value can refer to a distance that is beyond the end of the last parts of the Reply bundle, to defeat any estimation of the number of layers that it may contain.

↩️

Reply

In order to facilitate the return of an arbitrary blob of data as a reply to a message sent out by a client, there is a special construction of pre-made message which contains an header containing the forwarding and encrypted layers for the reply message.

To enable this, there is the Header/Payload key pair, the first is used on the header, and via the Offset field in

> In order to prevent the depth of the chain of forwards from being visible to relays, there must also be a random, arbitrary padding at the end of the header. Initially a rigid design was intended to cloak this, hiding the position on the path by it being moved upwards and padded back out for the next step, so a random length of padding that varies enough to make it difficult to know how many layers might be inside it must be used.
>
> This is now changed so that a random amount of padding is always added to the Header segment of a message. The padding should be filled with noise, a hash chain is sufficient and efficient for this.
>
> ↩️

Header

Byte length	Name	Description
2	Length	Length of Header, after which Extra Data is found
...	...	repeat 1 or more Forward, Crypt and then optionally Delay, Dummy and Pad layers

↩️

Extra Data

These are found directly appended to the end of the above header

Byte length	Name	Description
2	Count	Number of Symmetric Keys/IVs found in the following.

Byte length	Name	Description
32	Ciphers	Pre-generated symmetric keys created using the Payload session key and the same public key found in the matching Header layer (hidden from Exit/Hidden Service via encryption)
16	IVs	IVs that match the ones used in the header Crypt layers

⤴️

Ciphers

Sessions consist of two symmetric secret keys, the Header and the Payload keys. The Header key is used in the header shown above, to derive the Cloaked session header public key.

The Ciphers contain a series of symmetric secret keys that are the product of using ECDH on the one-time public key in the Crypt, and the session Payload key. The relay can thus encrypt an arbitrary message payload using this key

⤴️

Initialisation Vectors

The IVs used with the ciphers above, and wrapped in the Crypt messages, must be a separate set of IVs from the ones in the header. They must also be the same number as the Ciphers.

⤴️

Exit

Byte length	Name	Description
4	Magic	Sentinel marker for beginning/end of messages
8	ID	Database key for retrieving pending message
2	Port	Well known port *
...	Reply	Header, Ciphers and Initialisation Vectors for bundling Response
4	Length	Length of data following that contains request
...	Request	Message to forward to Service via local port forward

The Exit message is a request to tunnel a packet out of the Indra network.

Indra nodes advertise these as *Services*, which are identified by a Well Known Port, such as 80 for HTTP, 443 for HTTPS, 25 for SMTP, 22 for SSH, and so on.

Traffic that is forwarded to a *Service* is billed according to the average of the inbound message size and the outbound message size that is received back from the service in response to the request that was received inside the Exit message.

> * Well known ports are generally defined in unix for the privileged low ports, but also in protocols, such as 8333 for Bitcoin, 9050 for Tor, etc.

> A list will be compiled to add any extra features that are not services that would normally be defined as they are not normally public.
>
> For example, proxy port numbers are arbitrary, but we might specify they are to always be, say, 8080 for a HTTP proxy, or 8004 for Socks 4A or 8005 for Socks 5, and so on. These will most likely be the loopback ports that are usually used or even specified in the protocol, even though for clearnet use such an open relay would be a security/spam risk, this is the purpose of Indra, and spam is controlled separately by the metering of data volume for relaying

⤴️

Request

Byte length	Name	Description
4	Magic	`requ`
3	Length	24 bit value for the length of the message, i.e.: up to 16 Megabytes
...	Data	The payload of the Request

Request is exactly a standard request message for a server as found in all Client/Server protocols. It is simply a wrapper for an arbitrary payload of data.

Note that this, and the Response message are 24 bits and limit to 16 Megabytes in size. Messages containing payloads longer than this must be segmented at the application level.

⤴️

Response

Byte length	Name	Description
4	Magic	`resp`
3	Length	24 bit value for the length of the message, i.e.: up to 16 Megabytes
...	Data	The content of the Response

Response is the message wrapper for the response from a Service.

⤴️

Delay

Byte length	Name	Description
4	Magic	`dely`

Byte length	Name	Description
4	Delay	Number of milliseconds to hold message in cache before processing next message layer

In order to add some temporal jitter and to arbitrarily increase and decrease the size of the layers that are stacked at the head of messages, it is possible to add a specification that provides a millisecond precision 32 bit value specifying an amount of time to hold the message in the buffer before sending it.

In order to facilitate this the relays must charge according to a coefficient multiplied by the time of delays against the message size.

For this reason also, as will be explained with Pad, this is entirely under the control of the client for both reasons of securing anonymity as well as permitting applications to add these to messages for whatever purpose the application developer envisions, such as, potentially, storing data temporarily out of band for a prescribed amount of time as a form of "cloud storage".

⤴️

Pad (Increment/Decrement)

Byte length	Name	Description
4	Magic	`pado`
2	Difference	16 bit signed value that allows up to 32 Kilobytes of data to be added or removed from the message by the relay.

Pad is an instruction to append or trim bytes

Outside of the anonymity quality-of-service, reciprocating dummy traffic that peers will send to each other, using a scheme of fractional reserve for allowing temporal disjoint reciprocation, or whatever scheme ends up being used, the client needs to have control over how the size of their messages is altered deliberately in transit.

Failing to perform this is not detectable except in the last hop before the node as being the act of a given relay, but the client will recognise this as a sign that one of the hops in the path was processed not in accordance with the instructions and all nodes in the path will have a ban score increment the existence of which will prejudice greater ban scores if the error repeats in paths the relay is part of in future. (todo: pending responses must include all pad instructions in a message so this can be checked.)

⤴️

Dummy

Byte length	Name	Description
4	Magic	`dumm`
...	Destination	Forward header of destination for dummy message

Byte length	Name	Description
...	Crypt	Empty crypt

This message doubles the relaying fee for the outbound byte count of the message that was received, to be filled with randomly generated bytes, using a Hash Chain generator.

⤴️

Split

Byte length	Name	Description
4	Magic	`splt`
2	Count	Indicates the number of splits following
4	Offsets	*Count* number of 32 bit offset values marking the segments
...	Data	The segments that begin from the end of the Offsets/Destinations and continue until the end of the message.

Split is where the remainder of the message has 2 or more segments that each bear a header indicating a different destination. This could be used, for example, to create a liveness detection along an arbitrary route that conceals the return paths of this telemetry.

Splits also make possible the fan-out/fan/in pattern for multipath messages.

The actual instructions on where to forward the segments is at the head of each segment in cleartext.

⤴️

Low Latency Mixnet "Lightning Bolts"

One of the biggest difficulties with mixnets is that the lower the latency, the easier it is to correlate traffic paths as they flow through the network.

Defeating this attack can be achieved by adding Split messages fan out randomly Dummy , so that at each hop, at least two different simultaneous transmissions take place.

These forks should be constructed using Dummy packets after the Forward and Crypt messages in front of it, which may then be followed by further layers for second or third or further movements. In fact, Dummy padding can be used in addition to the Delay and Pad instruction messages as well to further break up the regularity of the sizes of forwarding segments.

These can be called Lightning Bolts since they propagate in a similar way as arcs of electricity across the sky and to the ground, forking towards equally conductive or from equally charged areas that merge or split.

The simulation of merging can even be created, as well, with forks that merge back together. This is achieved using Dummy messages.

⤴️

Network Intelligence

In source routing systems, the nodes that perform relaying services must advertise their existence and instructions on how to reach them.

All advertisements contain the following 4 fields, and additional fields as required:

⤴️

Ad Prototype

Byte length	Name	Description
4	Magic	`!!!!` (This item does not exist alone)
8	ID	Random value that ensures the signature is never placed on the same hash
33	Key	The public identity key of the relay providing relaying or other service
8	Expiry	The timestamp after which the ad must be renewed as all peers will evict the record from their network intelligence database
64	Signature	Schnorr signature that must match with the Key field above

These are the 4 essential elements in an ad, as shown above, and all the ads for both public and hidden services contain this. The signature, of course, is always at the end, but the order of the fields *could* be different.

⤴️

Peer

Byte length	Name	Description
4	Magic	`peer`
8	ID	Random value that ensures the signature is never placed on the same hash
33	Key	The public identity key of the relay providing relaying or other service
8	Expiry	The timestamp after which the ad must be renewed as all peers will evict the record from their network intelligence database after this time. This is both to keep the network metadata fresh, and to help keep cache sizes reasonable.
4	RelayRate	The price, in MilliSatoshi, for a megabyte of data
64	Signature	Schnorr signature that must match with the Key field above

Peer is simply the advertising of the identity of a peer. It contains only the public identity key, and the relay rate charged by it.

⤴️

Addresses

Byte length	Name	Description
4	Magic	`addr`
8	ID	Random value that ensures the signature is never placed on the same hash
33	Key	The public identity key of the relay providing relaying or other service
8	Expiry	The timestamp after which the ad must be renewed as all peers will evict the record from their network intelligence database
1	Count	Number of addresses listed in this, maximum of 256, 0 being the first
8/20/?	Addresses	Network Addresses - variable length. IPv4 and IPv6 encoding lengths with 2 byte port numbers added
64	Signature	Schnorr signature that must match with the Key field above

⤴️

Services

The first type of service provided over Indranet is public **Services**. These are services that are advertised by relays, that designate routes that messages to them can tunnel out to, outside of Indranet.

Byte length	Name	Description
4	Magic	`svcs`
8	ID	Random value that ensures the signature is never placed on the same hash
33	Key	The public identity key of the relay providing relaying or other service
8	Expiry	The timestamp after which the ad must be renewed as all peers will evict the record from their network intelligence database
2	Count	Number of services advertised, consists of Port and RelayRate
2	Port	16 bit port number of service, based on Well Known Port

Byte length	Name	Description
4	RelayRate	The cost in MilliSatoshi per megabyte of, the mean of request and response byte size
64	Signature	Schnorr signature that must match with the Key field above

⤴️

Hidden Services

Hidden services are a composition of Primitives that enables the initiation and message cycle of enabling bidirectional location obfuscation.

It borrows a little from the hidden service negotiation protocol used by Tor, but the connection is maintained by the hidden server and hidden client directly, bypassing the bottleneck and small attack surface created by a mediated rendezvous connection.

Overview

Prior to explaining the parts, it is necessary to list them, and this is best done as a numbered sequence, and the three parties involved we will use the common names used in cryptography that apply to this protocol.

Alice

Alice is the hidden service provider. Alice is generally the initiator in most scenarios described in cryptography.

Bob

Bob is the hidden client. Bob wants to talk to Alice, but doesn't know where she is picking up messages from.

Faith

Faith is often used as a trusted intermediary, however in this protocol she is serving in the role of an introducer, and her service is temporary.

Note that currently there is no scheme for billing hidden service traffic, essentially the cost of relaying is borne equally by the hidden service and the hidden client. Faith is essentially paid for this service as the hidden service must have a session with her to do this. Each time a new introduction is received, she is paid.

> note: in discussions of attacks on this protocol, the name **Eve** would be perfect as the placeholder for the attacker.

⤴️

Hidden Service Protocol

1. Alice wants to offer a hidden service, without disclosing to the network the location where the data is being processed or stored. This has especially got relevance to such services as trusted intermediaries, as a repository of secrets, even if encrypted, is a high value target for attackers.
2. Alice generates an introduction message, which consists of her hidden identity key, and the identity key of the chosen intermediary, Faith, is part of this message.
3. The second part of her message is secret, and consists of a **Reply** message, which will be used to forward a request to Alice. In order to prevent gathering any information about this return path, each one is single use, and after an introduction is consumed, the introducer waits for a new one.
4. The delivery of this introduction, and subsequent new Reply messages is done via an anonymised pathway that is typically 3 hops, but could be shorter or longer. *3 is just the magic number that is the maximum bang per buck for creating a path that can be difficult to trace, similar to how an extraction that takes 50% or better per pass exceeds 90% after 3. This is known as "The Rule of Three".*
5. Faith broadcasts this introduction across the network, in part because when hidden clients request a connection, she gets paid for forwarding the request.
6. Because Faith has a privileged position as the go-between, where she is doing a one-time version of the Tor hidden service, Alice will rotate the set of introducers, that is, Alice will send out many of these public/private intro/reply messages to peers, thus serving to create a moving target for would-be attackers, and also, on the other side, Faith is the attack surface for attempts to unmask the identity and location of Alice.
7. Bob receives the gossip about the various introducers related to the public identity key of Alice's hidden service, and wants to start a hidden conversation with her, without revealing his location either.
8. Bob sends a request to Faith via an obfuscated multi-hop path, containing a * *Reply** header, which is then forwarded back to Alice by Faith using the single use reply header she currently has for Alice.
9. Alice then receives this request, and then constructs a two or more layer forwarding prefix to add an extra two or more hops on top of the path that was given by Bob, the reason being that Bob might control the first hop, and be trying to unmask Alice. In this way, he would be thwarted at such an attempt. This reply is called **Ready**, and is like the Clear To Send signal on a serial connection, this step in the process akin to the handshake of TCP or similar in that now both parties are ready to start a conversation.
10. Bob then receives this message, which contains a Reply header from Alice, and Bob uses this, again with his own header prefix, to send a **Request** message, and wraps this in 2 or more hops to protect his location from the relay Alice provided in her **Reply**.
11. This then gets back to Alice, who can then forward the **Request** on to the actual hidden service server, who then returns a reply and Alice then wraps this in her two or more forward layers, around Bob's **Reply** header and inside that, the **Response** from the hidden service.

Protocol Messages for Hidden Services

Note that at all times, for reasons of eliminating the possibility of unmasking either end of a hidden connection, each side prefixes their messages with at least two hops in the path. This acts as a firewall against an evil counterparty controlling the relay at the top of the **Reply**.

Intro

Byte length	Name	Description
4	Magic	`intr`
8	Nonce	Random value that ensures the signature is always on a different hash
33	Key *	Public key of the hidden service identity, matching the Signature below
33	Introducer	Public key of the relay that is serving as the introducer
8	Expiry	The timestamp after which, by the clock of the introducer that the intro will be evicted from its network intelligence database
64	Signature	Schnorr signature on the foregoing message that matches **Key**

The intro is the publicly visible document that contains a signed designation of the introducer (Faith) in the protocol.

It is gossiped over the Publish/Subscribe system of Indra that propagates information about peers, their addresses, and their offered public services.

> * This key is encoded in the Indranet `Based32` encoding, which is 26 lower case Latin letters and 234679, which are the least ambiguous 6 numbers out of the 10 Arabic number ciphers. This custom encoding is used because it provides potential for later use of vanity addresses. But more than this... A note is required about this.

🔝

Rate Limiting Hidden Service Advertisements

Because there is no cost to generating, and a relatively low cost to publishing hidden services, in order to limit the amount of new hidden service addresses, they must contain a common 25 bit prefix that forms the word `indra`. Well, this is the provisional idea, it may need to be a longer prefix than this to be sufficiently limited in the context of possible ASIC devices for mining these keys. Currently the public key derivation operation is fairly expensive, very few Tor hidden services have more than 7 or 8 base32 characters forming their " vanity" prefix.

🔝

Introduction

Byte length	Name	Description
4	Magic	`inta`
150	Intro	The intro advertisement
...	Reply	The Reply message that the introducer returns a Route back with

Introduction is the message sent out by a hidden service, over a multi-hop path, to the Introducer they have chosen for a time to serve as introducer.

The Intro is gossiped via the Publish Subscribe peer to peer protocol, and every time a client sends a Route request, the hidden service will send a new one, as it will refuse to accept the second one of these.

This prevents a race condition and the possible plaintext attack that the same cipher set might open up.

⤴️

Route

Byte length	Name	Description
4	Magic	`rout`
8	ID	Random value used by the hidden client to identify the connection request
...	Reply	The path to send back the Ready signal

Route is essentially a connection request, after which the hidden client will expect to receive a Ready message containing a new **Reply** header to send a * *Request**.

⤴️

Ready

Byte length	Name	Description
4	Magic	`redy`
8	ID	The random value that was sent in the request is returned with the reply for quick retrieval
...	Reply	The **Reply** header the hidden client can use to reply

The ready message essentially functions in the same way as a standard handshake acknowledgement, which establishes that both sides are ready to begin a conversation, and the first request may be sent.

⤴️

Hidden Message Cycles

Aside from the unmasking prevention prefixes, and the **Reply** headers, the pattern of **Request** and **Response** are the same as any client/server protocol.

In order to maintain the connection, each side retains a number of prior **Reply** headers (3-5?) that were sent, and in the event of a transmission failure, the former, known successful **Reply** headers are retained for resending the message that did not get a reply. The TTL of such chatter is governed by the protocol that is being transported, on its "Application" layer, to use OSI layer cake nomenclature.

In the event that the client consumes all of its cached past **Reply** headers for the service, it can simply search out, or just use, other **Intro** advertisements that it has received over the gossip network, and reestablish the connection.

⤴️

Example Custom Protocols

> todo: some examples of other ways to use the primitives described in the foregoing, here are some headings:
>
> 1. Time Delay Data Storage
> 2. Metered Network Access
> 3. Paywalling access to content/databases/application
> 4. Additionally maybe discuss various strategies for defining paths, most especially, forks. Oh, I will discuss that in a section above this.

Glossary

⤴️

Based32

Based32 is a variant of Base 32 encoding which provides all letters of the alphabet in lower case to make it possible to have any human readable segment generated in a public key to assist with recognition and/or act as part of a proof of work style limit on generating hidden service identity keys.

The ciphers are the following:

```
abcdefghijklmnopqrstuvwxyz234679
```

This should be adequately resistant to transcription errors due to the relative differences between the ciphers as rendered in most fonts and handwriting. Only i, t and l are a little similar visually. Omitting capital letters sidesteps the ambiguity between Z and 2, 5 and S, 6 and G. g and 9 are similar in some renderings, as can be 9 and q, but 0, 1, 5, 8 are the most ambiguous numbers alongside lower case Latin letters.

⤴️

Client

Client is an application that constructs messages using this message format to have the data transported in the intended way across the network to its intended destination and back.

⤴️

ECDH

Elliptic Curve Diffie Hellman Key Exchange is a scheme that uses asymmetric (Public Key Infrastructure) cryptography to enable two parties to create a shared symmetric encryption cipher (for AES encryption) over an insecure channel, without enabling eavesdroppers or other attackers to acquire this secret and compromise the privacy of their messages.

⤴️

Header Key

Used for the construction of simple Forward, for Reply message construction, as contrasted with the Payload key, which is used for the separate segment of the message found after the Crypt Offset.

This is a 32 byte Elliptic Curve private key, used with other keys to generate symmetric encryption shared secrets using Elliptic Curve Diffie Hellman (ECDH) key agreement protocol, the same as the Payload Key.

Hash Chain

A hash chain is a sequence of bytes that starts with the system entropy generated 32 bytes, and followed by the product of hashing the previous hash, and so on, and then, usually, trimmed to the specified length. This is a lower cost random stream used to pad messages so that it is not possible to determine what is message and what is padding.

⤴️

Payload Key

After a Crypt Offset number of bytes, which is the first 3 bytes encrypted by the Header Key, this is the second key found in a Session, which enables the creation of a bidirectional message construction scheme that is controlled primarily by the client.

⤴️

Initialisation Vector

In order to prevent the repeat of generated encryption streams, every encrypted message must have a provided, cleartext prefix that is concatenated to the encryption secret to generate the beginning of the cipher stream/blocks.

⤴️

Fingerprint

A fingerprint is a shorter, usually derived value that makes it easier and faster to identify a longer string of bytes. An example of this is the RIPEMD160 hash of the public key associated with a Bitcoin address. In Indra this is generally done via SHA256 32 byte hash, that is truncated to a shorter length for the purpose. The Cloaked Key is an example of the use of this with a Nonce.

↩️

Nonce

A random value used for various reasons in cryptography. It is critical these values are generated by a Cryptographically Secure Pseudo Random Number Generator (CSPRNG) or entropy capture device, as patterns in these values can lead to plaintext cryptanalysis attacks that can uncover symmetric ciphers and/or private keys.

↩️

Relay

An Indranet relay is a server that accepts payments via the custom configured Indra Lightning Network sub-net, that has no routing fees due to the primary charging method of Sessions.

↩️

Sessions

A session is a pre-paid balance denominated in MilliSatoshi, 1/1000th of a Satoshi, the primary unit of the Bitcoin ledger. Sessions are created by making a payment via LN to the relay's identity key (one key that is used by the Relay, Lightning node and Bitcoin node) using Atomic Multi-Path payments, initiated by Clients and constructed so as to hide the origin of the payment.

↩️

Preimage

In the Lightning Network, payment contain a 32 byte hash value that associates with the payment a secondary piece of data used to prove/confirm the payment. In Indra, this data relates to the hash of the Header and Payload of a Session.

↩️

Magic Bytes

These are a string of plain ASCII characters (Latin) of 8 bits per character, and 4 bytes long to ensure that random byte sequences are unlikely to occur in the positions that the Magic should appear. Primarily used to indicate the beginning of a message, at the front of packets and directly after previous messages.

These enable fast recognition of the encoding to be expected, and the fields that will be found within a message segment by relays.

They are like the methods in an API, and are followed by the parameters that the API call requires.

↩️

Cloaked Key

A Cloaked key is a way of indicating the use of a specific Session's keys, specifically referring to the Header key.

This is generated using a random blinding factor, that is concatenated with the public key and appended to the end of the blinding factor, as follows:

```
fingerprint = hash ( nonce | public key ) -> truncated to 4 bytes

cloak = nonce | fingerprint
```

The vertical bar `|` represents the concatenation of the items on either side of it, for example: 123 | 4567890 = 1234567890

The relay can then scan its session database by generating the same construction using the same method just described to determine if the candidate key matches, as it iterates its Sessions database.

Cloaking prevents any relay other than the one that has the Session gathering any information about the identity of a session connecting two or more messages sent by the same Client.