ETSI GR IPE 012 V1.1.1 (2022-08)

IPv6 Enhanced innovation (IPE);
IPv6-based Blockchain

Disclaimer

The present document has been produced and approved by the IPv6 Enhanced Innovation (IPE) ETSI Industry Specification Group (ISG) and represents the views of those members who participated in this ISG.
It does not necessarily represent the views of the entire ETSI membership.

Reference
DGR/IPE-012

Keywords
bitcoin, blockchain, IPv6, security

ETSI

650 Route des Lucioles
F-06921 Sophia Antipolis Cedex - FRANCE

Tel.: +33 4 92 94 42 00 Fax: +33 4 93 65 47 16

Siret N° 348 623 562 00017 - APE 7112B
Association à but non lucratif enregistrée à la
Sous-Préfecture de Grasse (06) N° w061004871

Contents

Intellectual Property Rights

Essential patents

IPRs essential or potentially essential to normative deliverables may have been declared to ETSI. The declarations pertaining to these essential IPRs, if any, are publicly available for **ETSI members and non-members**, and can be found in ETSI SR 000 314: *"Intellectual Property Rights (IPRs); Essential, or potentially Essential, IPRs notified to ETSI in respect of ETSI standards"*, which is available from the ETSI Secretariat. Latest updates are available on the ETSI Web server (https://ipr.etsi.org/).

Pursuant to the ETSI Directives including the ETSI IPR Policy, no investigation regarding the essentiality of IPRs, including IPR searches, has been carried out by ETSI. No guarantee can be given as to the existence of other IPRs not referenced in ETSI SR 000 314 (or the updates on the ETSI Web server) which are, or may be, or may become, essential to the present document.

Trademarks

The present document may include trademarks and/or tradenames which are asserted and/or registered by their owners. ETSI claims no ownership of these except for any which are indicated as being the property of ETSI, and conveys no right to use or reproduce any trademark and/or tradename. Mention of those trademarks in the present document does not constitute an endorsement by ETSI of products, services or organizations associated with those trademarks.

DECT™, **PLUGTESTS™**, **UMTS™** and the ETSI logo are trademarks of ETSI registered for the benefit of its Members. **3GPP™** and **LTE™** are trademarks of ETSI registered for the benefit of its Members and of the 3GPP Organizational Partners. **oneM2M™** logo is a trademark of ETSI registered for the benefit of its Members and of the oneM2M Partners. **GSM**® and the GSM logo are trademarks registered and owned by the GSM Association.

Foreword

This Group Report (GR) has been produced by ETSI Industry Specification Group (ISG) IPv6 Enhanced innovation (IPE).

Modal verbs terminology

In the present document "**should**", "**should not**", "**may**", "**need not**", "**will**", "**will not**", "**can**" and "**cannot**" are to be interpreted as described in clause 3.2 of the ETSI Drafting Rules (Verbal forms for the expression of provisions).

"**must**" and "**must not**" are **NOT** allowed in ETSI deliverables except when used in direct citation.

Executive summary

Bitcoin introduced a breakthrough in peer-to-peer electronic cash and payments in 2008. This paradigm enabled payments to be sent inexpensively over the Internet, allowing peers to use IP addresses to communicate and transact directly between one another on the blockchain.

Despite its potential, this mechanism for direct payments using IP addresses was later deprecated due to contemporary vulnerabilities to attack that were associated with the IPv4-based Internet infrastructure of the time.

In the present document, the model for peer-to-peer payments over the Internet is revisited in the context of modern Internet protocols, such as DNSSEC and IPv6. An outline is given for how direct online payments can be reintroduced into the blockchain ecosystem by leveraging these secure modern technologies.

The present document shows how the properties of IPv6 lend themselves to new methods for constructing direct payments between Internet users, and how these same principles can be extended to new use cases for machine-to-machine and human-to-machine payments, taking full advantage of the expanded address space of IPv6 and the ability to associate Bitcoin payments with cryptographically-generated IPv6 addresses.

1 Scope

The present document outlines how the 'IP-to-IP' paradigm for payments can be reintroduced to the blockchain, and how the properties of IPv6 in particular can be leveraged to achieve new direct payment mechanisms for users of the blockchain.

2 References

2.1 Normative references

Normative references are not applicable in the present document.

2.2 Informative references

References are either specific (identified by date of publication and/or edition number or version number) or non-specific. For specific references, only the cited version applies. For non-specific references, the latest version of the referenced document (including any amendments) applies.

NOTE: While any hyperlinks included in this clause were valid at the time of publication, ETSI cannot guarantee their long term validity.

The following referenced documents are not necessary for the application of the present document but they assist the user with regard to a particular subject area.

[i.1] GitHub: "Remove send to IP address and IP transactions support" by laanwj Pull Request #253 bitcoin/bitcoin".

NOTE: Available at https://github.com/bitcoin/bitcoin/pull/253.

[i.2] Nro.net: "Regional Internet Registries | The Number Resource Organization".

NOTE: Available at https://www.nro.net/about/rirs/.

[i.3] IETF RFC 1883: "Internet Protocol, Version 6 (IPv6) Specification".

NOTE: Available at https://tools.ietf.org/html/rfc1883.

[i.4] Jwz.org: "The Rise of "Worse is Better".

NOTE: Available at https://www.jwz.org/doc/worse-is-better.html.

[i.5] IETF RFC 1034: "Domain Names - Concepts and Facilities".

NOTE: Available at https://datatracker.ietf.org/doc/html/rfc1034.

[i.6] Cloudflare: "How DNSSEC Works".

NOTE: Available at https://www.cloudflare.com/dns/dnssec/how-dnssec-works/.

[i.7] IETF RFC 5914: "Trust Anchor Format".

NOTE: Available at https://datatracker.ietf.org/doc/html/rfc5914.

[i.8] IETF RFC 3425: "Obsoleting IQUERY".

NOTE: Available at https://datatracker.ietf.org/doc/html/rfc3425.

[i.9] Simpledns.com: "DNSKEY-Records (DNSSEC public key)".

NOTE: Available at https://simpledns.com/help/dnskey-records.

[i.10] Simpledns.com: "RRSIG-Records (Rrset Signature)".

NOTE: Available at https://simpledns.com/help/rrsig-records.

[i.11] GitHub: "bitcoin-sv-specs/op-return".

NOTE: Available at https://github.com/bitcoin-sv-specs/op_return.

[i.12] Google.com: "Ipv6 - Google".

NOTE: Available at https://www.google.com/intl/en/ipv6/statistics.html#tab=ipv6-adoption.

[i.13] IETF RFC 3972: "Cryptographically Generated Addresses (CGA)".

NOTE: Available at https://datatracker.ietf.org/doc/html/rfc3972.

[i.14] École Polytechnique Fédérale de Lausanne (2009): "Analysis and Optimization of Cryptographically Generated Addresses (CGA) Revisiting Self-Certifying Address Generation and Verification".

NOTE: Available at http://secowinetcourse.epfl.ch/previous/08/Bos.Joppe_Ozen.Onur/Final_Report.pdf.

[i.15] IETF RFC 4581: "Cryptographically Generated Addresses (CGA) Extension Field Format".

NOTE: Available at https://tools.ietf.org/html/rfc4581.

[i.16] Qadir S. and Siddiqi M. (2011): "Cryptographically Generated Addresses (CGAs): A survey and an analysis of performance for use in mobile environment". IJCSNS International Journal of Computer Science and Network Security.

NOTE: Available at http://paper.ijcsns.org/07_book/201102/20110204.pdf.

[i.17] Cloudflare: "DNS A Record" Cloudflare.

NOTE: Available at https://www.cloudflare.com/learning/dns/dns-records/dns-a-record/.

[i.18] IETF draft-shen-csi-ecc-00: "Public Key Algorithm Agility in SEND".

NOTE: Available at https://tools.ietf.org/html/draft-shen-csi-ecc-00.

[i.19] IETF RFC 6605: "Elliptic Curve Digital Signature Algorithm (DSA) for DNSSEC".

NOTE: Available at https://tools.ietf.org/html/rfc6605.

[i.20] NIST Special Publication 800-57 Part 1: "Recommendation for Key Management: Part 1: General", Revision 5, 2020 E. Barker.

NOTE: Available at https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-57pt1r5.pdf.

[i.21] Web.archive.org: "Bitcoin v0.1 released".

NOTE: Available at https://web.archive.org/web/20110309003215/https://www.mail-archive.com/cryptography%40metzdowd.com/msg10142.html.

[i.22] IETF RFC 4301: "Security Architecture for the Internet Protocol".

NOTE: Available at https://tools.ietf.org/html/rfc4301.

[i.23] Merkle R. (1978): "Secure Communications Over Insecure Channels". Communications of the ACM. 21 (4): 294-299.

NOTE: Available at https://dl.acm.org/doi/10.1145/359460.359473.

3 Definition of terms, symbols and abbreviations

3.1 Terms

Void.

3.2 Symbols

Void.

3.3 Abbreviations

For the purposes of the present document, the following abbreviations apply:

ACME	Automated Certificate Management Environment
AH	Authentication Headers
BIP	Bitcoin Improvement Proposal
BSV	Bitcoin Satoshi Vision
CA	Certificate Authority
CGA	Cryptographically Generated Address
CGN	Carrier-Grade NAT
DNS	Domain Name System
DNSKEY	DNS Signing Public Key Record
DNSSEC	Domain Name System Security Extension
ECDSA	Elliptic Curve Digital Signature Algorithm
ESP	Encapsulating Security Payloads
GPG	Gnu Privacy Guard
IP/TCP	Internet Protocol/Transport Control Protocol
IPv4, IPv6	Internet Protocol (version 4, version 6)
ISAKMP	Internet Security Association and Key Management Protocol
KINK	Kerberized Internet Negotiation of Keys
MITM	Man-In-The-Middle
NAT	Network Address Translation
P2PKH	Pay to Public Key Hash
PGP	Pretty Good Privacy
PK	Public Key
PKI	Public Key Infrastructure
PTR	Pointer
rDNS	reverse DNS lookup
RFC	Request For Comments
RIPEMD160	RIPE Message Digest (160-bit)
RRSIG	Resource Record SIGnature
RSA	Rivest-Shamir-Adleman public key cryptosystem
SA	Security Associations
SHA256	Secure Hash Algorithm (256-bit)
SSL	Secure Sockets Layer
TCP	Transmission Control Protocol
TFL	Transport For London
TLS	Transport Layer Security
Tx	Transaction
TXT	Text Record
UTXO	Unspent Transaction Output
VPN	Virtual Private Network
ZK	Zone Key

4 IPv6-based Blockchain: Overview and Requirements

4.1 Introduction

From its inception, Bitcoin was designed to be compatible with persistent routable IP addresses to enable IP-to-IP (IPv4-to-IPv4, without NAT or CGN in between) transactions built into the source code, see [i.21]. The way such an IP transaction works is as follows:

- The client contacts the recipient's IP address to find out if they are running Bitcoin and accepting IP transactions. If not, no transaction occurs.
- Additional information, such as "from", "message", etc., can be included with this.
- The recipient server generates a new, one-time Bitcoin public key and sends it to the client.
- The client sends the payment to the recipient's one-time public key.

This method, while functional, provided no authentication on the recipient's public key and was therefore vulnerable to a Man-In-The-Middle (MITM) attack. In such an attack, a malicious MITM could intercept the message and have the payment sent to themselves instead during the above process, especially if the recipient is using a proxy server and fraudulently poses as the intended recipient while providing the sender with their own Bitcoin address instead.

Towards the end of 2011, support for IP transactions was removed from the Bitcoin codebase [i.1] due to its perceived vulnerability to such attacks. However, with the introduction of network infrastructures such as Domain Name System Security Extension (DNSSEC) and Certificate Authorities (CAs), most of the internet traffic in the present day is, in fact, authenticated (and, by extension, encrypted).

The present document, therefore, proposes that IP-to-IP transactions may be reintroduced into Bitcoin and other blockchain networks in such a way that it is invulnerable to MITM attacks by leveraging these modern advances in network infrastructure. For example, certificates are used to ensure the authenticity of the public key associated with an IP address and provide a record of the communication in an OP_RETURN statement that can be monitored by the server.

In clause 4.2.3, IP transaction solutions utilizing the IPv4 standard are considered. This clause describes how IPv4-to-IPv4 Bitcoin payment protocols can be facilitated by leveraging the existing DNSSEC and TLS certificate frameworks to generate payment addresses from IPv4 addresses.

Subsequently, in clause 4.2.4, these protocols are further developed to exploit the advantages offered by using IPv6 addresses in lieu of IPv4. These enhanced protocols accommodate a variety of methods for sending IPv6-to-IPv6 Bitcoin payments, including the use of cryptographically generated addresses that can have the dual purpose of securing an IPv6 address and receiving payments.

Following this, in clause 4.2.5, a privacy extension that obfuscates payment keys from exchanged messages is introduced. In clause 4.2.6, a number of different use cases where these techniques can be utilized are detailed.

4.2 Preliminaries

4.2.1 The Internet Protocol Suite

The Internet Protocol Suite is the set of communications protocols that integrate seamlessly through different layers during internet communications and interactions. It is also commonly referred to as the TCP/IP Layer Model because the suite comprises two foundational protocol layers, namely the Transport Control Protocol (TCP) and the Internet Protocol (IP). In this model, there are four main layers: Network Access, Internet, Transport and Application, as shown in figure 1.

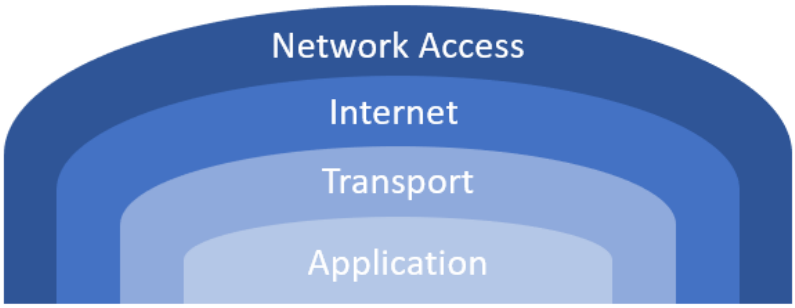

Figure 1: Internet Protocol Suite Layer Model

Note that the Internet layer is often referred to as the IP layer, and the Network Access layer is often split further into the Data-Link and Physical layers.

These layers are analogous to the layers of an onion, whereby the Network Access layer is the outermost layer, and the Application layer is the innermost component. These layers progressively build upon and encapsulate each other, which means the Transport layer encapsulates the Application layer, and similarly, the Internet layer encapsulates the Transport layer. Internet data packets are first assembled according to the above layers and then sent over the internet. Along the way, the data is unpacked or the layers 'peeled-off', but only the layers are required to route the specific packet to its destination, where the host unpacks the whole packet.

There are two major standard implementations that are widely used to facilitate the IP Layer, namely version four (IPv4) and version 6 (IPv6). Earlier versions 0-3 and 5 were experimental versions. When IPv4 was first deployed in the 1980s, the creators did not anticipate the extremely widespread use of the internet. As a result, only 32-bits were allocated for the size of the IP addresses, which limits the space of possible IPv4 addresses to approximately 4,3 billion unique addresses. This may have seemed sufficient at the time, but, given the subsequent growth in Internet usage, this is increasingly not the case. As a result, combined with the limitations of IPv4 addresses, there are currently just five regional internet registries that allocate these addresses to different parts of the world [i.2].

To solve this problem, IPv6 was introduced, with larger 128-bit addresses, in 1995 (IETF RFC 1883 [i.3]). However, by this time, the wider internet usage has become largely predicated upon the IPv4 standard, which has since made it difficult to implement widespread upgrading to the usage of IPv6. As a result, IPv4 addresses are now often re-used and remapped through the use of Network Address Translation (NAT) and Carrier-Grade NAT (CGN) to stay backwards compatible with IPv4. NAT is not an ideal solution for a number of reasons and brings with it its own issues, but it is nonetheless a ubiquitous part of the network infrastructure of the Internet, see [i.4].

The Domain Name System (DNS) is a hierarchical naming and discovery system used for Internet resources [i.5]. It essentially maps human-readable domain names, such as "google.com", to machine-readable addresses, such as 8.8.8.8, and vice versa. There can be many different types of DNS records associated with a single given domain. Some common types include records for IP addresses (`'A'` for IPv4 and `'AAAA'` for IPv6), pointers for reverse DNS lookups (`'PTR'`), and records for DNSSEC (`'RRSIG'` and `'DNSKEY'`). Example DNS records can be seen in [i.17].

4.2.2 Internet Protocol Security and Extensions

4.2.2.1 Introduction

When the Internet Protocol Suite was designed, it did not necessarily take security into great consideration and was left vulnerable to some forms of exploitation. This has resulted in numerous hacks and attacks that took advantage of its security inadequacies. To mitigate this, security extensions and new protocols were invented, with examples shown below. It is noted, however, that some of the security additions were not ideal as they sacrificed security for usability to remain backwards compatible. This is especially demonstrated with IPv4 and expanded on further below.

4.2.2.2 Domain Name System Security Extension (DNSSEC)

The DNSSEC is a suite of the Internet Engineering Task Force (IETF) specifications that secures DNS records and information through the use of Certificate Authorities (CAs) and a Public Key Infrastructure (PKI). Cloudflare released an informative article that explains how DNSSEC works on a technical level, see [i.6].

4.2.2.3 Transport Layer Security (TLS)

The TLS protocol, and its deprecated predecessor, Secure Sockets Layer (SSL), are cryptographic protocols that facilitate encrypted Internet communications from the Transport layer inwards using CAs and a PKI (similar to that of DNSSEC).

The handshake protocol, used by a **Client** and a **Server** to establish a secure communication channel, involves the following steps (shown in figure 2):

1) The **Client** sends a `client_hello` message to the server, which includes a version number, its supported cypher suites, and extra options.

EXAMPLE: TLS_DHE_RSA_AES_256_CBC_SHA.

2) The **Server** responds by sending a `server_hello` message (similar to the one sent by the **Client**).

3) The **Server** also sends the key-exchange information, depending on the options selected (usually either RSA or DH_RSA or DHE_RSA).

4) The **Server** also sends its SSL/TLS certificate (issued to it by a CA).

5) The **Server** can also request a certificate from the **Client** but does not always need to.

6) The **Client** sends its certificate if it was requested.

7) The **Client** does the required cryptographic computations needed and sends back its key-exchange information.

8) The **Client** sends `change_cipher_spec` notification and changes its record layer security state to symmetric encryption using the freshly created session keys from the above key-exchange information.

9) The **Server** responds with the same `change_cipher_spec` notification.

10) The **Client** and **Server** then communicate and exchange application data over the secure channel established. All communication sent between the **Client** and the **Server** is encrypted using the session key derived above.

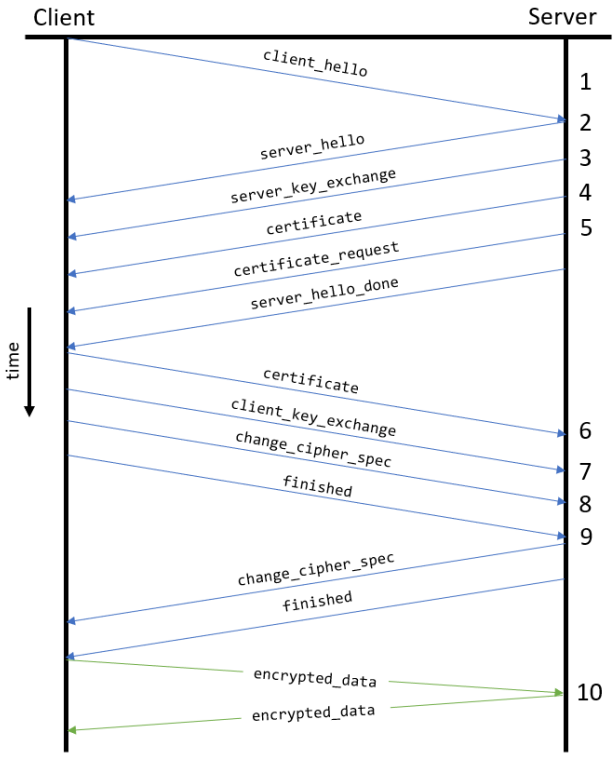

Figure 2: SSL/TLS Handshake Protocol

The SSL and TLS protocols rely heavily on CAs and a PKI to provide authentication between the two communicating parties and protect against MITM attacks. However, as mentioned above, IPv4 addresses suffer from a major limitation in terms of the space of possible addresses, which necessitates the re-use and remapping of addresses. Consequently, SSL/TLS certificates are issued on domain names instead of IPv4 addresses because they are more specific and permanent.

Towards the end of the 2000s (2000-2010), much more elegant solutions to the PKI problem, such as clause 4.2.2.4, were proposed using IPv6, but by that time, it was already too late. Most of the Internet was already running on IPv4, and it was hard to get everyone to upgrade to use IPv6. Even though the introduction of non-ideal solutions like NAT has delayed IPv6 adoption, IPv6 adoption is steadily increasing over time, as can be seen in the data on IPv6 usage provided by Google® [i.12].

4.2.2.4 Cryptographically Generated Addresses (CGA)

A Cryptographically Generated Address is a self-certifying address and is used as a method for binding public keys to an IPv6 address without the need for a CA or PKI. It is an IPv6 address that is cryptographically derived from a public-private keypair. The address is cryptographically linked to a public-private keypair, so anyone who verifies the correct generation (self-certification) of the CGA can be guaranteed a certain degree of security that the link is valid. How this works technically is explained further below.

The basic principle involved in achieving this is shown in [i.14]. The most significant 64 bits of the IPv6 address are reserved for the subnet and are referred to as the subnet prefix. The least significant 64 bits are generated from a cryptographic hash of the public key of the address owner and are referred to as the interface identifier.

This simplified CGA scheme had its security-enhanced by introducing two different hashes and a 'proof of work' method (similar to Bitcoin mining) to make it much harder for an adversary to attack, which was proposed in IETF RFC 3972 [i.13]. This relies on a *Sec* parameter that the first hash is based upon. The higher the value of this security parameter, the more difficulty of the first hash is required (similar to bitcoin mining difficulty). This makes address generation and verification more complex, essentially presenting a trade-off for security that sacrifices efficiency. This updated mechanism for cryptographic address generation is outlined in [i.14].

An extension field format was introduced in IETF RFC 4581 [i.15], which allows for extending the hash output for higher security and less collision probability. More detailed flow charts for generation and verification of CGAs can be found in [i.16].

4.2.2.5 CGA++

CGA suffers from a couple of limitations that are not extremely significant in network communications, especially when a hash extension is used. Of these limitations are the garbage attacks, replay attacks, or the time-memory trade-off attacks, which are mostly impractical and not very useful attacks in communications since the attacker will only have a public key that maps to the IPv6 address but no private key corresponding to it to mount any useful attacks. This, however, is not insignificant when looking to use the IPv6 identifier to send a Bitcoin transaction.

To address these limitations, a new protocol based on CGA, denoted by CGA++, was proposed in 2009 [i.14]. The main difference stems from the introduction of a signature by the private key into the hash function used to generate the IPv6 address. In other words, authentication is incorporated into the address generation and verification, as opposed to outside with CGA (to set up communications, a signature is sent along in parallel to set up the secure channel). This further decreases efficiency, but not by a significant amount as demonstrated in the results of [i.14].

4.2.2.6 Comparison between CGA and CGA++

The primary difference between CGA and CGA++ is that authentication under CGA++ is done inside the address generation (and, by extension, verification). This security trade-off comes with an extra cost, however in terms of added complexity in address generation and verification. However, this increase in complexity does not have an actual practical impact that compares the CGA and CGA++ for IPv6 using a 1 024-bit RSA key [i.14]. This protocol was first proposed in around 2009, so with the technological improvements made since then, address generation and verification should be even less expensive now. However, it is noted that even though CGA++ is an improvement on CGA, it still does not have an RFC of its own.

4.2.2.7 Internet Protocol Security (IPsec)

IPsec is an internet suite protocol that was added in IETF RFC 4301 [i.22] to authenticate and encrypt packets from the IP layer inwards. IPsec is used in Virtual Private Networks (VPNs) to extend private networks across the public internet. IPsec works with both IPv4 and IPv6. It is an optional setting with IPv4 however a mandatory one with IPv6.

The IPsec suite consists of three main protocols: Security Associations (SA), Authentication Headers (AH), and Encapsulating Security Payloads (ESP). The SA protocol is used to provide the bundle of algorithms and data exchanges necessary for AH and/or ESP protocols. AH is used to guarantee authentication and integrity of the data sent. ESP is used to provide what AH provides, in addition to the confidentiality of the data.

Since most of the internet uses the legacy framework of IPv4, many of the protocols used to establish Security Associations are within that framework. The framework for establishing SAs is provided by the Internet Security Association and Key Management Protocol (ISAKMP). This includes protocols such as the Internet Key Exchange (IKEv2), which can use X.509 certificates (same as SSL/TLS), Kerberized Internet Negotiation of Keys (KINK), which uses a trust third party, and configurations which are done manually. IPv6 CGAs help to mitigate some of the complexity and inefficiency that are associated with Public Key Infrastructure.

4.2.3 IPv4-to-IPv4 transactions

4.2.3.1 Introduction

In this clause, IPv4-to-IPv4 transactions using version 4 of the IP layer (IPv4) are considered, and a new protocol provided for the secure sending of blockchain payments to an IPv4 address. The case where the receiver possesses some form of certification or authentication through a Certificate Authority (CA) and in a Public Key Infrastructure (PKI) is also considered. Certificate authority chains of trust within a PKI provide considerably high-security guarantees that act as trust anchors for applications [i.7]. Typically, with IPv4, this authentication can be done through two primary methods: DNSSEC and SSL/TLS. The process is described in detail below. However, the primary goal is to use the authentication provided through DNSSEC or SSL/TLS to guard against a MITM attack, such as the one described in the introduction.

4.2.3.2 Method Using DNSSEC Authentication

This clause details the procedure needed for a **Client** to send a payment directly to a recipient, which may be represented by a **Server**, a domain name or an IP address. The only prerequisite is the assumption that the recipient has a domain name and is certified using DNSSEC. For additional security, the receiver can set up a DNS `TXT` record that indicates that they are indeed using this protocol (note that a DNS `TXT` record can contain any extra information pertaining to the owner of the domain).

If the Client already knows the domain name it wants to send a payment to (instead of just an IP address), skip to step 2). Otherwise, assume the Client only knows the IP address.

1) Client issues a reverse DNS (rDNS) [i.8] lookup for the domain name associated with the specified IP address by querying for the `PTR` record.

2) Client sends a query for the `DNSKEY` [i.9] record to get the domain's Zone Key, ZK and check that they are indeed signalling the use of this protocol.

   If DNSSEC is **not** being used, **STOP**.

   - There is no proper authentication being done.
   - It is not secure to send money to this domain.

3) **Verification:** to verify the public key (ZK) in the certificate is associated with the target domain name, the client requests the `Rrset` and the `RRSIG` records (see [i.10]). The `Rrset` is basically the set of records associated with the domain (for example, `A` or `AAAA`). The `RRSIG` record is the signature on this data using the `DNSKEY`, which is the public key of the certificate authority.

NOTE 1: The recommended settings for DNSSEC signing are ECDSA with a 256-bit key (curve p-256) as opposed to RSA, so the private/public keys are natively compatible with Bitcoin cryptography [i.19] and [i.20].

4) Client calculates the new public key $P_{IP} = ZK + SHA256(M) \times G$

The message M may be anything related to the payment, such as the concatenation of the IP address and the Unix time $M = (IP\ address||unix\_time)$ or alternatively the invoice information of payment.

NOTE 2: The procedure can be thought of as a "one-sided" version of a secret value distribution protocol, where the server's master key $P_{MS} = ZK$, and the procedure ends once the client calculates the server's paired public key $P_{2C} = P_{IP}$.

5) Compute the HASH160 of the public key and then get the base58 encoding of that to get the P2PKH value to send the payment to.

6) Send the desired amount in a transaction to the above P2PKH address $H(P_{IP})$ including an additional OP_RETURN output with the following information in the locking script (figure 3).

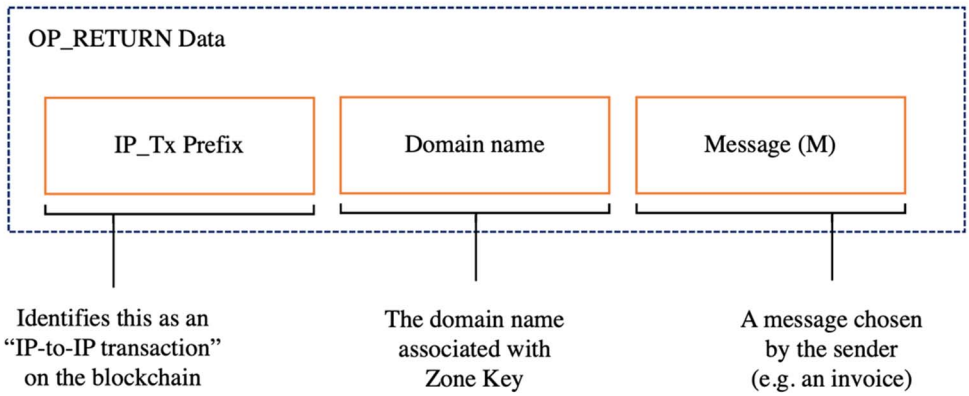

Figure 3: OP_RETURN Output Data

NOTE 3: This can be standardized in the central repository for various OP_RETURN protocol specifications [i.11].

7) Finally, the Server can monitor the blockchain waiting for payments made to its domain in the OP_RETURN field. When a payment is detected, the Server can use its private key (v_{ZK}) corresponding to the zone public key, ZK, to calculate the private key (v_{IP}) for the received UTXO sent to P_{IP}. The equation to calculate $v_{IP} = v_{ZK} + SHA256(M)$.

NOTE 4: This procedure is non-interactive, so the receiver does not need to be online to receive the payment.

$TxID$	
Inputs	**Outputs**
$<Sig\ P><P>$	`OP_RETURN <IP_Tx prefix> <Domain_Name> <M>`
	`OP_DUP OP_HASH160 <`$H(P_{IP})$`> OP_EQUAL OP_CHECKSIG`

Figure 4: Example Non-Interactive Bitcoin IPv4 Transaction

The main advantage of using this mechanism is that it is non-interactive, and no extra functionality needs to be implemented except to check for payments made to a given recipient on the blockchain (by checking the OP_RETURN field with the `IP_Tx` prefix). Web servers will usually be online, but they will need to add this extra functionality to send addresses to new customers. With this process, this is not needed.

4.2.3.3 Method Using SSL/TLS Authentication

If the receiver does not have a domain name with DNSSEC integrated, authentication can still be achieved through SSL/TLS (not easily). Even though SSL/TLS certificates are almost always issued on domain names instead of IP addresses, it is still possible to have an SSL/TLS certificate on an IP address. Most SSL/TLS certificates on the internet are issued for free through an automated service provider, which use Automated Certificate Management Environment (ACME) challenges. However, SSL/TLS certificates on IP addresses cannot be done for free using automated services because there is no way to properly authenticate the IP address. Thus, to get an SSL certificate on an IP, it can be purchased from a provider that supports issuing certificates on public IP addresses, as many do not. This is understandable as IPv4 has too few addresses than are needed in the world, so many are interchanged and not permanent. If the receiver is willing to go through all this trouble, they might as well get a domain name, which would probably be even easier. On the other hand, for local solutions, it is possible to create self-signed certificates which can be utilized in internal systems such as companies or organizations.

4.2.3.4 Inherent IPv4 Limitations

This is the main issue with the problem of authentication that is trying to be solved with this whole procedure. Possible solutions to this problem hit a brick wall because of the nature of the way the system works, specifically Ipv4. If the internet infrastructure were to move to use IPv6 instead of IPv4, this would create the opportunity to allow new solutions to such a problem. Using IPv6 rather than IPv4 immediately addresses the PKI issue for routing and, by extension, could address the central issue of this whitepaper when linking an IP address to a public key (and thus a blockchain address). Taking a step back and approaching the situation from outside of the tunnelled point of view of IPv4, there are much better solutions when using a better version of the Internet Protocol (IPv6).

4.2.4 IPv6-to-IPv6 transactions

4.2.4.1 Introduction

Limitations with Public Key Infrastructure (PKI) have existed since the early days of the Internet. These limitations on are extended to blockchain networks naturally such as blockchains such as Bitcoin currently run on top of the Internet. Once the majority of the world transitions to using IPv6 instead of IPv4, more elegant solutions become possible and can be used to solve many of these problems. As shown in [i.12], IPv6 adoption is steadily increasing, indicating that such widespread IPv6 usage may be on the horizon.

4.2.4.2 IPv6 Transactions Using CGA(++)

It is clear that Bitcoin payments are a natural fit with Cryptographically Generated Addresses due to the issues discussed regarding IPv4. With the use of IPv6 Cryptographically Generated Addresses, authentication can be done easily as a feature of the way CGAs work. This authentication can be extended easily to derive a session key (through Diffie-Hellman or RSA) to provide confidentiality in a secure channel between sender and receiver, similarly to how TLS works, see clause 4.2.2.3. As mentioned in clause 4.2.2.7, there is no need for traditional PKI and Internet Key Exchange when using CGA, as that was the whole purpose of creating CGA. End-to-end authentication and encryption can be done using IPsec; thus, there is effectively no need for TLS.

With this framework in place, sending and receiving Bitcoin IP-to-IP transactions directly, as originally intended for the Bitcoin protocol, can be done securely. This can be easily added into user software to check if the receiving host is using IPv6 and then to enable Bitcoin transactions. The procedure is as follows:

1) The client contacts the recipient's IP address to find out if they are running Bitcoin and accepting IP transactions. If not, no transaction occurs.

2) Additional information, such as "from", "message", etc., can be included with this.

3) The recipient server generates a new, one-time Bitcoin public key address (P2PKH) and sends it to the client. This bitcoin address can be a new, completely random one or can be generated deterministically using the BIP 32 protocol or using another secret value distribution protocol. This can also be a transaction template created by the recipient or a custom locking script that the recipient generates and sends to the client.

4) The client creates the payment to the recipient's one-time public key, or in the case where the sender receives the transaction template, the client signs that and sends it back to the recipient, who settles that on the bitcoin blockchain.

4.2.4.3 Non-Interactive IPv6 Transactions Using CGA(++)

It is important to emphasize that the procedure below is a modification of clause 4.2.3.2 in a way that takes advantage of the beneficial properties of IPv6 (and CGA), which are not possible with IPv4. To send a Bitcoin transaction to a recipient who is not online (non-interactively), the sender can take advantage of the embedded authentication, using a signature, in CGA++ (as opposed to CGA, where authentication happens externally). In addition, the sender can just re-use CGA parameters if having communicated with the recipient in the past.

The procedure to send a Bitcoin transaction to a recipient CGA++ is as follows:

1) Client determines the address to send the payment to (CGA++ along with parameters, including public key PK).

2) Client verifies the address is properly generated.

3) Client calculates the new public key $P_{IP} = PK + SHA256(M) \times G$.

   The message M may be anything related to the payment, such as the concatenation of the IP address and the Unix Time, $M = (IP\ address||unix\_time)$, or alternatively the invoice information of a payment.

NOTE 1: As a result of the method being proposed early in the past before elliptic curve cryptography was common, the public key PK used with CGA was defined as RSA in IETF RFC 3972 [i.13] however, there is a public key extension that can be added to upgrade the CGA (and SEND) protocols to incorporate the use of elliptic curve cryptography, see [i.18].

4) Compute the HASH160 of the public key and then convert to the base58 encoding to obtain the P2PKH destination to send the payment to.

5) Send the desired amount in a transaction to the above P2PKH address $H(P_{IP})$, including an additional OP_RETURN output with the following information in the locking script (figure 5).

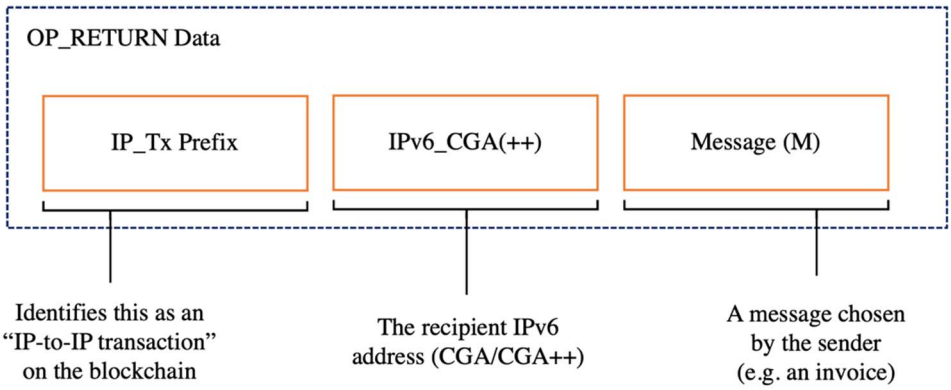

Figure 5: OP_RETURN Output Data

NOTE 2: This can be standardized in the central repository for various OP_RETURN protocol specifications [i.11].

6) Finally, the Server can monitor the blockchain waiting for payments made to its domain in the OP_RETURN field. When a payment is detected, the Server can use its private key (v_{PK}) corresponding to the CGA++ public key, PK, to calculate the private key (v_{IP}) for the received UTXO sent to P_{IP}. The equation to calculate the corresponding private key needed to redeem the funds is simply $v_{IP} = v_{PK} + SHA256(M)$.

$TxID$	
Inputs	**Outputs**
$<Sig\ P><P>$	OP_RETURN <IP_Tx prefix> <IPv6_CGA(++)> <M>
	OP_DUP OP_HASH160 $<H(P_{IP})>$ OP_EQUAL OP_CHECKSIG

Figure 6: Example Non-Interactive Bitcoin IPv6 Transaction

4.2.4.4 Domain Name Bitcoin Payments Using IPv6 CGA(++)

To send a payment to a domain name rather than an IPv6 CGA(++), the sender simply adds an extra step at the beginning of the procedure to resolve the IP address that maps to the domain name required. In the CGA parameters, there is an *extFields* parameter where an optional variable-length field (default length 0) can be added. If the receiver/Server has a domain name, they can put that there so that the Client has added security guarantees that the DNS mapping is authentic (check step 3). The protocol to do this is described below:

1) Client issues DNS query to resolve the IPv6 (`AAAA` DNS record) address mapping to the required domain name.

2) Verify DNS record(s) (if DNSSEC is being used).

3) Check that the domain name in the DNS record matches the domain name in the *extFields* CGA parameter.

4) Procedure continues as in clause 4.2.4.2 or clause 4.2.4.3.

NOTE: Usually, the security of DNS when using DNSSEC relies completely on the chain of certificate authorities and the public key infrastructure. However, with the addition of the extFields CGA parameter, authentication is further guaranteed.

4.2.5 Extensions

4.2.5.1 Privacy Extension

The non-interactive Bitcoin IP transaction method described in clauses 4.2.3.2 and 4.2.4.3 lacks privacy. Any unrelated observer can just check the blockchain and see the `OP_RETURN` output message and link it to the transaction sent. However, this can be avoided by slitting the outputs into two different transactions (see figures 7 and 8) and encrypting the message such that only the sender and receiver can decrypt it from the blockchain. This can be done using a non-interactive version of the Diffie-Hellman key exchange algorithm [i.23]. The sender uses the public-private keypair P_1, v_1 corresponding to the UTXO being spent (using the first UTXO if there is more than one), as well as the receiver's CGA IPv6 public key, i.e. $IPv6_{CGA(++)} = PK$, to calculate a shared secret session key, k. The sender then encrypts, using a suitable symmetric encryption scheme, the message M using this session key k. The session key can then be derived by both sender and receiver as follows.

$$P_1 = v_1 \times G$$

$$IPv6_{CGA(++)} = PK = v_{PK} \times G$$

$$k = v_1 \times PK = v_{PK} \times P_1 = v_1 \times v_{PK} \times G$$

$TxID_1$	
Inputs	**Outputs**
$< Sig\ P_1 > < P_1 >$	`OP_RETURN <IP_Tx prefix> <PK_Identifier> <Enc`$_k$`(M)>`

Figure 7: Private Bitcoin IPv6 Tx1

NOTE: The <PK_Identifier> field can be set to the <IPv6_CGA(++)> or the <Domain_Name> depending on which method is being used, namely clauses 4.2.3.1 and 4.2.4.2.

$TxID_2$	
Inputs	**Outputs**
$< Sig\ P_2 > < P_2 >$	`OP_DUP OP_HASH160 <`$H(P_{IP})$`> OP_EQUAL OP_CHECKSIG`

Figure 8: Private Bitcoin IPv6 Tx2

The receiver can then calculate the session key, k, themselves, decrypt the message M, and calculate the public key, P_{IP}, which the transaction output is sent to. Any unrelated observer will not be able to link the two transactions $TxID_1$ and $TxID_2$ to each other since they will not be able to decrypt the message M.

$$k = v_{PK} \times P_1$$

In fact, the <PK_Identifier> field can be generalized to any kind of public key or public key identifier, and the scheme can be extended for any type of use case and is not constrained to just IP transactions and/or messages.

4.2.5.2 Certificate Authority Extension

The above methods consider some of the public key infrastructures or certificate authorities that are available at the time of writing. In the case of IPv4, DNSSEC and TLS/SSL are used. In the case of IPv6, CGAs are used. However, the methods discussed could be implemented given any other public key infrastructure techniques. The main requirement is that there is a trusted link between an IP address and a specific public key. If the public key is not an ECDSA public key, a hash function can be applied to map it to an ECDSA public key. Once a trusted link is established, then sent to an IP address becomes as secure as sending to a trusted public key.

4.2.6 Use Cases

4.2.6.1 Webserver Access

Bitcoin IP transaction functionality can be easily leveraged using webservers today, enabling them could integrate this functionality to charge clients for access. For example, a webserver could request a certain payment to reply to the client who is requesting certain access or functionality to a website or webserver. This payment could also be turned into a payment channel to allow for micropayments to be made.

4.2.6.2 Local Link Payments

A situation where IP transactions would be useful is when making a payment to someone or in the immediate vicinity, namely on the same local network link. To find IP addresses on the same subnet, the sender can broadcast a ping request on the subnet and get replies from active IP addresses. Also, with IPv6, Secure Neighbour Discovery (SEND) protocol could be used to discover other network nodes on the local link. The sender can then make a Bitcoin IP transaction using whichever method of the ones described in the previous clauses to the receiver. An interesting example where this could be used is if a user wanted to easily pay someone who was physically close to them.

4.2.6.3 IP Messaging

Since anything can be included in the message M, this system can be used for any type of messaging, such as email. The procedure is identical to the procedure in clause 4.2.4.2 with the only difference that in the OP_RETURN output, the message M is replaced by the encryption of M using the recipient's public key, as shown in figure 9.

The message can be encrypted using the same method explained in the privacy extension in clause 4.2.5.1. The message can also be encrypted using Pretty Good Privacy (PGP)/ Gnu Privacy Guard (GPG), which uses a hybrid of asymmetric and symmetric encryption. This could be relevant since some email users already use PGP/GPG to encrypt and sign emails.

NOTE: This scheme is not constrained to just IP transactions and can be extended to any situation where the sender has a public key belonging to the receiver and wants to send them a private non-interactive payment.

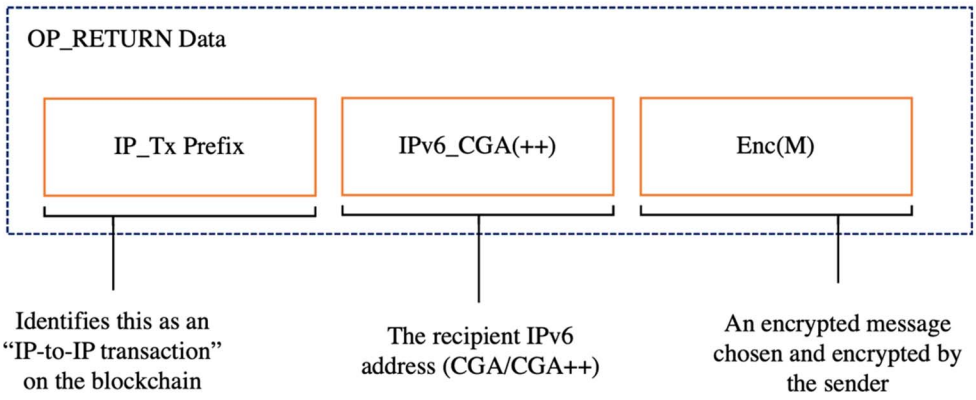

Figure 9: The on-chain data components of an example on-chain messaging scheme

4.2.6.4 Public Transport Terminals

The concept of making Bitcoin payments directly to an IP address is particularly powerful for scenarios where automated tasks are performed by electronic devices that each has an associated IP address.

A good example of this is public transport terminals, which are typically electronic barriers that are automated to permit a traveller to pass through if they provide the requisite payment. If an IP address can be associated with such a terminal, or a collection of terminals, then the methods outlined in clause 4.2.4 can be deployed to use IP-to-IP Bitcoin payments to authorize passage through the terminal(s).

For this example, assign the domain name '`tfl.gov.uk`' to the entity Transport For London (TFL), which could be responsible for the London Underground, and which has the corresponding IP address IP_{TFL}. Each station operating as part of the system may also have its own domain name '`station.tfl.gov.uk`' and IP address IP_S which represents the set of gate terminals at that station.

Assume that the TFL accepts Bitcoin payments for journeys. These payments can either be made prior to taking the journey or at the time of travel. The interaction between the traveller and a gate terminal will depend on which of these options is chosen.

In this example use case, a traveller, Alice wants to use the London Underground. Two scenarios are considered; firstly wherein Alice knows which station(s) she plans to use, and secondly, whereby Alice does not plan her journey ahead of time.

4.2.6.5 Scenario 1 - Prepaid journeys

In this scenario, assume that Alice plans her journey from station 1 to station 2 ahead of time. This means that Alice can pre-pay for her journey and receive two access tokens, which will allow her to pass the gate terminals at stations 1 and 2, respectively. The phases of her journey are as follows:

1) Prior to her journey, Alice broadcasts a transaction (figure 10) that pays TFL for her journey, on the condition that the TFL can provide two secret values X_1 and X_2 both chosen by Alice.

   This transaction is paid to the public key:

   $$P_{IP} = PK_{TFL} + SHA256(M) \times G\,,$$

   where PK_{TFL} is the certified public key associated with TFL's IP address IP_{TFL}, and the message M is the information detailing Alice's planned route, as well as a nonce to avoid collisions.

   The transaction's spendable output includes two hash puzzles, which means that Alice effectively only pays for her journey once it has occurred. Otherwise she can reclaim her funds (assuming an appropriately time-locked refund transaction has also been constructed). The solutions to these hash puzzles are Alice's secrets X_1 and X_2. These secrets will be used as the access tokens used by Alice at the gate terminals in stations 1 and 2, respectively.

It is assumed that this transaction is mined and visible to the London Underground's terminals, or the servers controlling them before Alice arrives and reaches station 1.

$TxID$	
Inputs	**Outputs**
$<Sig\ P_A><P_A>$	OP_RETURN <IP_Tx prefix> $<IP_{TFL}>$ <M>
	OP_SHA256 $<H(X_1)>$ OP_EQUALVERIFY OP_SHA256 $<H(X_2)>$ OP_EQUALVERIFY OP_DUP OP_HASH160 $<H(P_{IP})>$ OP_EQUAL OP_CHECKSIG

Figure 10: Alice's payment IPv6 transaction to TFL

2) When Alice reaches a gate terminal at her starting station, station 1, she simply sends the first preimage X_1 directly to the terminal itself.

   The terminal is able to verify that $H(X_1) = SHA256(X_1)$, which means that X_1 is indeed a valid access token for this station and allows Alice through the gate.

3) When Alice reaches a gate terminal at her end station, station 2, she sends the second preimage X_2 directly to the terminal.

   The terminal is able to verify that $H(X_2) = SHA256(X_2)$, that X_1 and X_2 now both known to the TFL. This is enough information for the TFL to consider Alice's journey paid for, and she is allowed through the gate.

Note that it may be sensible to relate the secret values X_1 and X_2 in such a way that makes the verifications done by the terminal more efficient. For example, choosing X_2 and $X_1 = H(X_2)$ as the secret values and rewriting the locking script as the script below would achieve this.

Locking script:

OP_SHA256 OP_DUP $<H(X_2)>$ OP_EQUALVERIFY OP_SHA256 $<H(X_1)>$ OP_EQUALVERIFY

Unlocking script:

$<X_2>$

If Alice pre-pays for her journey but chooses not to take it or takes an alternative route, she can still reclaim her funds using the corresponding refund transaction, as she will not have revealed both X_1 and X_2 to TFL terminals. In this case, Alice will simply default to scenario 2, which are described in full in clause 4.2.6.6.

4.2.6.6 Scenario 2 - Pay-as-you-go journeys

In this scenario, assume that Alice has not planned her journey in advance and instead wishes to be flexible with which journey she pays for. This means that the first terminal she reaches will construct a template transaction and give her an access token X_1 to use at the next terminal, she reaches. For the remainder of this clause, transaction values are given in units of BSV (the native token of the BSV blockchain) as an illustrative example, without loss of generality.

1) When Alice reaches a gate terminal at her starting station, station 1, the terminal generates a template IPv6 transaction and hands this, along with a token X_1, to Alice.

$TxID$			
Inputs		**Outputs**	
Value	**Script**	**Value**	**Script**
y	$<P_A>$	0	OP_RETURN <IP_Tx prefix> <station1.tfl.gov.uk> $<X_1>$
		x	OP_DUP OP_HASH160 $<H(P_{IP})>$ OP_EQUAL OP_CHECKSIG

Figure 11: Template IPv6 transaction generated by a gate terminal at station 1 and sent to Alice

This transaction is paid to the public key:

$$P_{IP} = PK_{S1} + SHA256(X_1) \times G\ ,$$

Where the subscript 'S1' denotes information related to station 1. Here, the value of the transaction x BSV is the minimum fare for a journey.

Alice responds by signing the transaction and giving the signature $Sig(P_A, Tx)$ back to the terminal. The terminal allows Alice through the gate but does not broadcast this transaction to the network, instead of sending it to the other terminals on the London Unground network.

2) When Alice reaches a second terminal at a second station, station 2, she provides the secret preimage X_1 and the appropriate signature. Next:

   a) If the journey fare f is exactly x BSV, the gate simply verifies that $H(X_1) = SHA256(X_1)$. If TRUE, the gate considers her journey paid and lets her through.

   b) If $f > x$ BSV, the gate gives Alice an updated template to sign (figure 12).

$TxID'$			
Inputs		**Outputs**	
Value	**Script**	**Value**	**Script**
y BSV	<P_A>	0	OP_RETURN <IP_Tx prefix> <IP_{S2}> <station2.tfl.gov.uk> <station1.tfl.gov.uk>
		$x < f < y$	OP_SHA256 <$H(X_1)$> OP_EQUALVERIFY OP_DUP OP_HASH160 <$H(P_{IP})$> OP_EQUAL OP_CHECKSIG

Figure 12: Updated template IPv6 transaction generated by a gate terminal at station 2 and sent to Alice

Note that the terminal at station 2 can calculate the fare f in-situ, as Alice's origin point (station 1) has been publicly broadcast and mined into the blockchain.

Alice provides the appropriate signature for this transaction $Sig(P_A, Tx')$ and returns this to the gate terminal.

The terminal now considers Alice's journey paid and lets her through the gate.

Features and benefits

- **Instant settlement of journeys**. Journeys are considered paid at the final terminal rather than being settled at a later time. This gives users more certainty over the amount of remaining funds.
- **Traceable journeys**. Journey information is encoded in the transaction messages, which means that a dispute over which route was taken, and paid for, should not arise.
- **Device-agnostic**. Use of signatures, based on private keys, can be made inter-operable between devices. The terminals will not perceive a difference between different devices used at different terminals. The TFL do not currently support device-switching during journeys.
- **No accidental payments**. The use of digital signatures makes it is impossible to accidentally pay for somebody else's journey, which is commonplace on the Underground currently.
- **Scenario-switching.** It is possible to begin a journey using scenario 1 and later change to scenario 2 simply by getting the second terminal to generate a template transaction for the user to sign if the scenario is switched.

  In general, scenario 1 may be encouraged using additional incentives that lower fares e.g. a yearly subscription/rail-card/season ticket.

4.2.7 Conclusion

Bitcoin was initially designed with consideration for native IP-to-IP transactions. Two years after its inception, IP transaction functionality was removed from the codebase by the Bitcoin core developer group. The issue with Bitcoin IP transactions is not an issue inherent to Bitcoin, but an issue with the way the Internet Protocol has historically functioned, namely IPv4 and legacy Internet infrastructure.

The present document proposes solutions to the issues present when attempting to send and receive Bitcoin IP-to-IP transactions. The ideal solutions revolve around the use of IPv6. However, it is still important to include possible IPv4 solutions since IPv6 usage is still only represents approximately 40 % of traffic, see [i.12].

In the context of IPv4, it is proposed to use Zone Keys (ZKs) to derive public keys (PKs) for receiving Bitcoin in an IP-to-IP transaction framework. Here, the solution relies upon DNSSEC and TLS/SSL certificates for mitigation of man-in-the-middle attacks and to safely derive payment addresses.

These concepts were extended to support and leverage IPv6 addresses, which yield significant advantageous properties over IPv4 when implemented. Specifically, it is proposed to use cryptographically-generated IPv6 addresses, whereby the native Public Key (PK) used in IPv6 address generation is also used for generating Bitcoin addresses. Here, the role of DNSSEC and TLS/SSL certificates are replaced with IPsec, which is already an inherent part of CGA protocols.

Novel methods for embedding the data relating to IP-to-IP transactions within the transactions themselves are also outlined, which allow a recipient to receive payment and construct the requisite private key using a form of non-interactive Diffie-Hellman key exchange.

History

Document history		
V1.1.1	August 2022	Publication