A. An Addressable Encrypted Blockchain

A.1. Overview

The Addressable Encrypted Blockchain (AEB) is the interaction of three different data contexts:

A.1.1. Address

The Address context delineates structure. It defines the extent of an AEB structure as well as allows users and other interacting resources the ability to work with individual blocks and structures of blocks.

By applying the Address context in a hashed tree structure, in the same vein as current Internet (IPv4/v6) addressing, it gives applications the ability to treat the entirety of an AEB network as a single, logically accessible, data structure. Every participating user or node no longer requires an entire copy of the AEB history, and individual blocks and structures of blocks on the AEB are available as though they are files on a filesystem or web addresses on the Internet.

A.1.2. Encryption

The Encryption context integrates all the resources which comprise and interact with an AEB structure. Based upon the Double Ratchet Protocol's encryption algorithm, it creates a common encryption context for all authenticated users and other authenticated interacting resources without exposing private key information.

From this context a unique encryption key is created for each individual block. Each authenticated participant generates each key locally from the common context and never actually share or distribute these keys.

A.1.3. Validation

The Validation context tests and verifies the AEB structure holistically as well as each block individually. Based upon the Blockchain's Merkle Hash, it uses Cryptographic Hash Functions (CHF) to prove that any given data block is a verifiable, unfalsifiable descendent of its parent, creating the same immutable journal that powers these existing Blockchain structures.

A.2. Together

These three technologies provide a baseline for everything one needs to build data management structure which is cryptographically controlled, cryptographically validated, and cryptographically secured. Encryption keys delegate authority over and availability for address blocks; each block is encrypted with keys that are unique and non-determinable outside of the delegated authority; and, the data itself is validated using well established Merkle Tree methodology.

The Addressable Encrypted Blockchain (AEB) is a product which merges all of these data contexts into a single, easy-to-use structure that simplifies and “abstracts away” all of the complexities involved. It can act, simultaneously, as a database and data storage system, a communications platform, a distributed computing platform, an application development platform, an intelligence collection platform, and an immutable transaction and event ledger.

A.3. Cryptographic Core

The core behavior of the AEB is built upon its cryptographic inspiration: the Double Ratchet Protocol. The simplest way to describe it is, after an initial handshake protocol between participants, referred to as a “shared secret”, each message is encrypted with a different temporary key, with each participant providing the other with enough information to generate, via a deterministic function called a Key Derivation Function (KDF), the next key.

A.3.1. Ratcheting Contexts

This back-and-forth advancement is why it’s referred to as a “ratchet”. What makes it a “double ratchet” is that each participant uses each message to both create a new shared secret and then use it to independently generate its next keys. The primary benefit of ratcheted or evolved keys is even if an attacker manages to decrypt one message, he does not have enough information to decrypt any other message of the conversation, since shared secrets are not themselves transmitted.

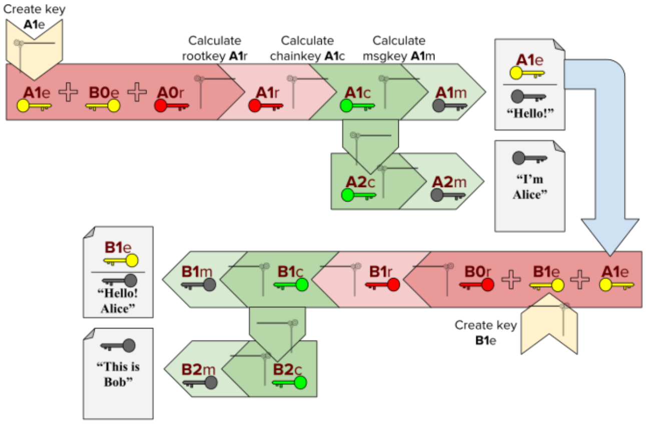


Fig A.2.1-1 Simplified Workflow of Cryptographic Double Ratchet.
Ephemeral keys are created and used to generate a ‘shared secret’ and Root key, then Chain key and Message Key

The initial handshake and “shared secret” creation, a new Cryptographic Context is created, and it is from this context that new data is appended to the chain with new, evolved key. In the Double Ratchet Protocol, this is called the Primary and Secondary Ratchets and these ratchets are triggered upon receipt of a new remote message and every subsequent sent message, respectively.

In the AEB, these trigger points are tied to the Addressing system and governed by configuration or whichever application owns the Cryptographic Context. We refer to them instead as the Root Context and the Block Context.

A.3.2. Root Context

The purpose of the Root Context is two-fold: to synchronize cryptographic context between participant nodes, and to spawn off synchronized child contexts, such as the Block Context.

A.3.2.1. Synchronization

In a traditional Double Ratchet scenario, synchronization occurs between two users and is handled via a Diffie-Hellman key exchange. The Diffie-Hellman handshake creates the shared context via the "shared secret" from which the child contexts and their keys are generated.

However, in a heterogeneous network environment with many Resources interacting with each other, there will be more than two participants with access to many Documents, so an AEB Document needs the ability to retrieve the shared secret via additional means.

By delegating the handshake to a plugin architecture called the Function Map Application, or FMApp, keys can be cached or created by other means and accessible by referring to an AEB Document by Address. Root Contexts can have their keys distributed to them by an "upstream" resource, generated on their behalf by an application-specific FMApp, or devolved to them by another Resources' handshake.

A.3.2.2. Child Contexts

Functionally, Root Contexts are created by applying a new Shared Secret to the previous Root Context, in the process extending a cryptographically valid chain of Root Contexts. This Root Context is itself a special function, when invoked, emits an extended key string which can be used as a genesis key for an arbitrary number of new, child, contexts.

A.3.3. Block Context

The Block Context uses said genesis key to create the individual Block keys using a Key Derivation Function (KDF). This KDF is deterministic, meaning that the same input will always produce the same output. In this case of the Block Context, this is the next Block's encryption/decryption key.

A.3.4. Messages and Blocks

In the context of the AEB, it's worthwhile to note that the ratchet of new keys is, like the Blockchain, also an example of Directed Graph, which means both technologies are categorically similar and highly compatible. The ratchet exchange is substantively similar to a block hash, and the inclusion of block hashes as an input to a ratchet's KDF means messages and blocks are essentially the same structure for these purposes.

Together, this means not only is the validity of a message-block assured via hash, and that an attacker who has compromised a transaction unable to decrypt an entire conversation-chain, a feature known as Forward Secrecy, but also any attempts to modify or corrupt a transaction with false message-blocks are immediately identifiable and can be transparently mitigated.

A.3.5. Key Infrastructure

Public Key Infrastructure (PKI) is essential and integral to the AEB and all of it's functionality. The Double Ratchet Protocol operates between two nodes just fine without validation or authentication, but doing real work in an evolving key architecture requires a fully versed PKI.

A.3.5.1. Hierarchy

The AEB, having a hashed addressing architecture, also accordingly has a hierarchical ownership architecture. Just as all Blocks are cryptographically valid descendents of a global genesis block, so too must all permissions keys descend from a universal genesis key.

It is this key, and the hierarchy of keys cryptographically chained or descended from it, that provides resources the authority to interact with other resources. Therefore, PKI functionality is built into the AEB in the form of special Function Map Applications (FMApps) which perform typical PKI and Certificate Authority functions like Creating Keys, Signing Keys, and Validating Keys.

A.3.5.2. Proof Of Ownership

The use of encryption also establishes proof-of-ownership for both message-block resources and for Document resources. Unlike Bitcoin's use of an expensive proof-of-work algorithm to confirm new blocks and build consensus among node resources, only the holders of the correct key(s) have the authority to advance the conversation-chain and append to a Document. With shared or group keys, block confirmation is restricted to trusted node resources, as is block decryption and chain assembly.

Additionally, with Public Key Infrastructure (PKI) these capabilities can be distributed based upon architecture and application needs. For example, remote nodes can have permissions to add message-blocks, but rights to canonically confirm blocks to the conversation-chain given only to trusted, central nodes, with an entirely different set of processing node resources given decryption keys.

A.3.5.3. Consensus and Merge

With ownership, consensus, for most network node resources, becomes a function of merely validating ownership for a block and a chain. This separation of authority allows trusted node resources to implement an application's own logic for chain extension.

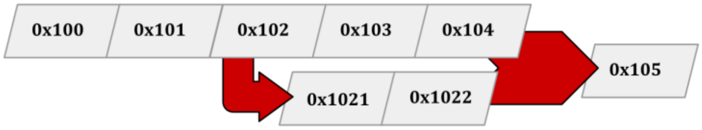


Fig A.2.1.3-1 Merger of Child Blockchain Back Into Parent Chain

Unlike Bitcoin's "longest chain" validation algorithm, otherwise orphan chain branches, created by multiple writer nodes and consisting of unconfirmed candidate blocks, can still be confirmed and a merge algorithm executed to create a new block containing branch data, as appropriate. The Git distributed version control system operates on this principle, and branched version control for the AEB is an additional benefit.

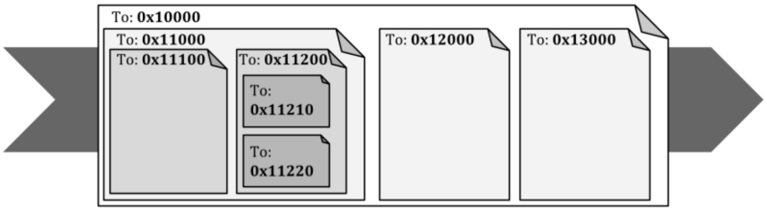


Fig A.2.2-1 Nested Encrypted Message-Blocks In Flight

A.3.6. "Onion Plus" Layered Encryption

The transfer of data between resources, both node and non-node, is comprised of blocks and blockchains just like the resources themselves. As such, blocks in transit between nodes themselves are comprised of blocks nested and addressed for their respective destination resources. This is analogous to the "onion" routing of Tor, where packets are layered with different destination addresses at each layer, ultimately obfuscating the communication.

Any observation of a blockchain shows only a nested stack of encryption, with each layer performed by a prior owner using his own secret key. If an attacker manages to decrypt a block, in flight or on disk, he still will not have enough information to decrypt any of the nested blocks contained within.

A.3.7. Assurance Hashing

Cryptographic hashes have been used for decades to track alterations to documents: if even one byte has changed, the calculated hash will differ dramatically from its expected value. In a blockchain, the determined hash value of a Block, generated from both the block and a secret key used by the owner, is embedded inside a subsequent Block. This allows us to verify the history of edits to a document, and to assure it's content.

A.3.7.1. Merkle Hashing

The hashing systems of Blockchain systems, including the AEB, is based upon the concept of the Merkle Tree. A Merkle Tree is a mathematical tree-and-leaf structure where each Block, acting as a "leaf", carries with it, or is "labelled", with the collective cryptographic hash of the history of Blocks that precede it, as parents, grandparents, etc. This allows any validation function the ability to prove that a given Block appropriately belongs to a Document or Blockchain.

Cryptocurrencies, like Bitcoin, are built around this feature. Blocks are composed of multiple transactions and are assembled, using cryptographic hashes as links, into a "chain" structure: the Blockchain. Combined, these linked blocks become an immutable ledger, a complete history of the transactions stored in that blockchain are available for review.

The AEB uses two different hash values to complete this capability: a Content Hash, and a Running Hash.

A.3.7.2. Content Hash

The Content Hash is the Cryptographic Hash value generated from a Block's unencrypted contents. It is used as the base for several different hash calculations. With the Content Hash, a Block's post-decryption status can be validated. This behavior is the same as how Cryptographic Hashes have been used for decades to validate, detect, and track alterations to documents: if even one byte has changed, the calculated hash will differ dramatically from its expected value.

A.3.7.3. Running Hash

The Running Hash is a Cryptographic Hash value generated from the history of Content Hashes in a blockchain or Journaled Data Document. It's primary purpose is to guarantee whether a Block is a valid child within its Document context; i.e. that it's a real child block and not an impostor. In the AEB context, this value is used for pre-validation before attempted decryption, as an additional security measure. If the Running Hash for a given Block is not valid within its Document, then the AEB will not waste system resources attempting to decrypt the Block and reject it instead.

A.3.8. Data Versioning

Other data resources, such as files or database documents, can also be represented in a Blockchain. Stored as one or more initial blocks, subsequent data changes can be assembled into descendent blocks, validated by the chain of cryptographic hashes. This is essentially how version control systems (VCS) used by software programmers work. The entire history of that data resource is available and can be further modified with new blocks appended to the chain.

A.4. Hashed Subnet Addressing

Traditional Blockchain systems, as well as traditional CDR-based messaging platforms, each presume that all participants operate in a single, contained, logical data structure. Bitcoin, for example, allows all network participants to examine all the transactions recorded by all users. Additionally, the next-generation Instant Message (IM) app Signal, the archetypal platform for CDR-based messaging, presumes that both participants have full access to all the content generated in a session.

The AEB presumes that neither of these scenarios is guaranteed. Therefore, in order for a User, Node, or Application resource to operate on the Block resources and Document chain resources it has key-defined permissions for, it is necessary that these resources be addressable within an overall common structure.

Just as internet resources are identifiable and addressable by their IP address, and file pieces in the BitTorrent file sharing system are addressable by downloaders, so too are Blocks and Document chains addressable in the AEB. In fact, it is addressing which turns the novelty of the Blockchain and the utility of the CDR into a general platform for assured application development.

This addition of Addressing to the AEB allows Resources to be addressed at both the macro-level, as in Network transport structures or 'Big Data' processing scenarios, as well as the micro-level, as in individual data tuples within a Document. This means that any application built with the AEB can simultaneously and separately interact with data it has the appropriate keys to, regardless of it's location on the AEB network, disparate on another Node, or local within memory.

A.4.1. Subnetworking

Drawing from the example of IPv4, where CIDR and internetworking rules allow hierarchical separation and special subnets and addresses, so too does the AEB use subnetworking to provide special features.

A.4.1.1. Hierarchical Subnets

Just as IP CIDR subnets are assigned hierarchically, so too are AEB subnets assignable hierarchically, albeit with a far larger address space (Currently proposed at 320 bits vs IPv4's 32 bits and IPv6's 128 bits). This allows controlled distribution of address space, as well as allowing subnet owners the authority to operate and distribute tenant subnet space under their own rules.

Addresses and Address Ranges are distributed in the form of a Structured Document, where each child address range is mapped to the Public Key of a generated key pair.

A.4.1.2. Cryptographic Key-Based Resource Ownership

Since ownership and authority in the AEB is designated via cryptographic key and operations on resources in a subnet's address range requires delegation also by cryptographic key, being addressable is not the same as being accessible. Unlike with IP networking, where routing and access rules are determined by multiple complex later protocols which operate independently, all access is controllable on an owned subnet, including total firewall for internal resources or total opacity for external resources.

A.4.1.3. Special Subnets

Just as IP address space has special reserved CIDR subnets and addresses, so too does the AEB. Special Document chain resources exist in these spaces, performing such tasks as network control, routing rules, resource metadata, authority and permissions distribution, block copy distribution, and key storage. These 'metachains' allow subnet owners the ability to securely and distributively control their subnets and the data structures represented within them according to their own configuration rules or application logic.

A.4.2. Scalability

The AEB address space is designed to function effectively at scales both macro and micro, while providing Bittorrent-like features such as speed, distribution, and redundancy. It can transparently and seamlessly function as versioned file storage, a distributed clustered file system, or store individual database tuples on a local disc, limited only by configuration or application logic.

A.4.3. Multiple Chains

sAddressing allows for multiple, distinct, chain resources to exist on a common network and operate under common platform rules. For example, Document chain 0x100 can begin at Block address 0x100 and contain Blocks from address 0x100 through, 0x120, while Document chain 0x200 begins at Block address 0x200 and contain Blocks from address 0x200 to 0x206. Under this scenario, Bob, who created and operates Document 0x100 would essentially ignore Document 0x200.

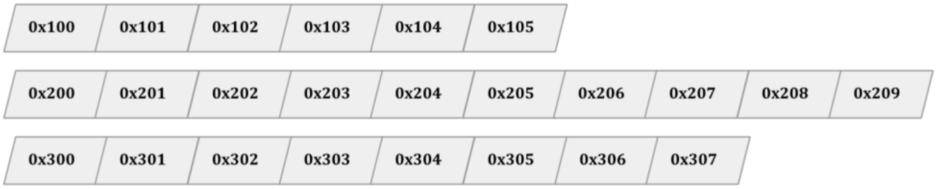

0x100 0x101 0x102 0x103 0x104 0x105
0x200 0x201 0x202 0x203 0x204 0x205 0x206 0x207 0x208 0x209
0x300 0x301 0x302 0x303 0x304 0x305 0x306 0x307

Fig A.3.1-1 Visualization of Segmented Chain Addresses

Depending upon keys granted by Alice, creator of Document 0x200, Bob may or may not perform limited functions for the Document, such as distribution and caching, but otherwise the Document is opaque to him. The two Document are distinct and operate under different application logic.

A.4.3.1. Complex Data Structures

Multiple distinct Document chains opens up the ability to operate complex data structures in a single platform. For example, a Document chain resource representing an event log could operate under a different set of rules, accepting data from Bob, Alice, and Charlie equally, than a Document chain resource representing a shared data file where Block resources from Charlie have merge priority and Alice doesn't even have the permissions to read. Unlimited multi-chain scenarios are supported, in the form of Structured Documents, limited only by the application developer.

A.5. Resources

All data interactions in the AEB are expressed as either a block-message Block resource or a conversation-chain Document resource (or a type derived from one of these). This includes cryptographic keys, owner-trust chains, ephemeral transactions, networks and subnets, resource metadata, logic and executable code, and change-deltas to these.

All primary resources are fully addressable network participants created as tenant subnet chain resources or tenant addressed block resources. For example, just as the Massachusetts Institute of Technology owns the 3.0.0.0/8 IPv4 block and control all addresses between 3.0.0.0 and 3.255.255.255 so might Bob, assigned an AEB subnet beginning with address 0x100 control all addresses between 0x100 and 0x199. Additionally, a chain beginning at block 0x100 would contain data necessary to identify Bob as the owner of the range, among other network-related metadata.

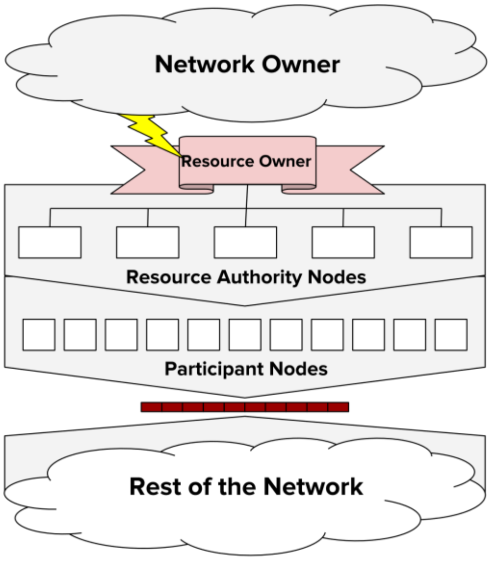


Fig A.4.1-1 Distribution of Authority for Resources.
Note the “wall” preventing non-participant nodes from interacting.

A.5.0.1. Cryptographic History

One of the guiding principles of an AEB is universal confirmed assurance. To achieve this standard, every resource must have at least one cryptographically verified parent, and every resource smust cryptographically descend from a “genesis” block. This chain of validation ensures that any resource actor of any type is a valid and confirmed network participant, that any and all data can be cryptographically traced to its validated source, and any attack vectors are siloed into controllable, and subsequently mitigatable workflows.

A.5.0.2. Metadata and Authority Distribution

Since primary resources are allocated by subnet block, as mentioned previously, special subnets are also designated for the storage and management of resource metadata via Structure Documents. Using the example of Bob’s 0x100 allocated subnet, addresses 0x100 through 0x109 are reserved for subnet metadata. Through the use of shared and/or group keys, authority over this resource metadata, including tenant subnet resources, can be designated to and distributed among any other resources of the subnet owner’s choice.

A.5.1. Resource Authority (RA)

The network effects of addressability and the cryptographic control of resources allow application developers to assign arbitrary distribution of authority within a AEB subnet. With resource permissions distributable by key, the AEB includes common functional logic, in the form of a built-in FMApp, for permitted node and code resources to mediate actions between resource actors.

Using permission keys generated and assigned by the resource owner, any node or code resource can act as a full or partial authority (Resource Authority, RA). Using group keys and multiply-signed keys, multiple resources can be designated to perform RA functions, allowing for economies of scale for high-volume applications. RAs operate within 5 broad categories of behavior:

A.5.1.1. Cryptographic Key Operations

RAs with cryptographic permissions have the ability to create keys, generate private/public key pairs, generate shared secrets between handshaking resources, sign keys, confirm keys and key signatures, delegate trust functions, and perform other cryptographic functions as needed.

A.5.1.2. Block and Chain Operations

RAs with Block and Chain permissions can confirm or reject blocks for a chain, conduct merge operations, validate and compile chains, manage block redundancy and distribution, chain consensus validation, and perform other chain management functions as needed.

A.5.1.3. Network Control and Interoperation

Network RA functions include creation and control of tenant subnets, maintenance of message-block routing maps, control of interactions with peer and parent subnet RAs, maintenance of bootstrap chains for new node participants, and validation and inclusion of new node participants.

A.5.1.4. Resource Metadata Management

These functions include creation, destruction, and maintenance of resource metadata chains, including but not limited to keychains, resource event chains, tracking chains, private chains, chains internal to RA operation, and resource permissions chains. Additionally, these functions provide and store resource metadata and transparently make such data available to other RA functions, including data compilation, decryption, local storage and configuration rules.

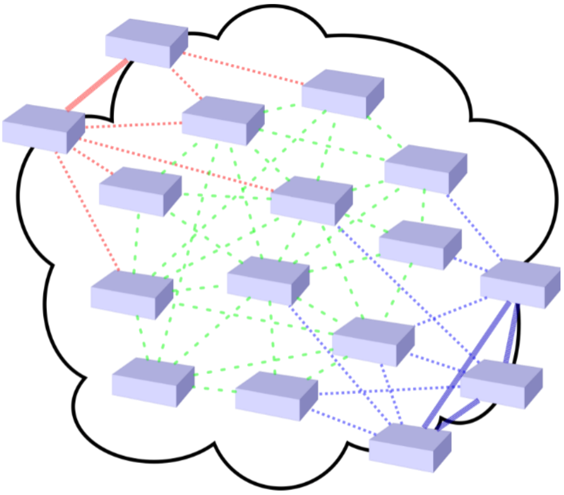


Fig A.5.1-1 A Mesh Network. Note that each node is connected to several other nodes.

A.6. Network and Communications Layer

Fundamental to the operation of the AEB is the network of node resources which operates it, and that operation is governed by the following principles.

A.6.1. Mesh Network

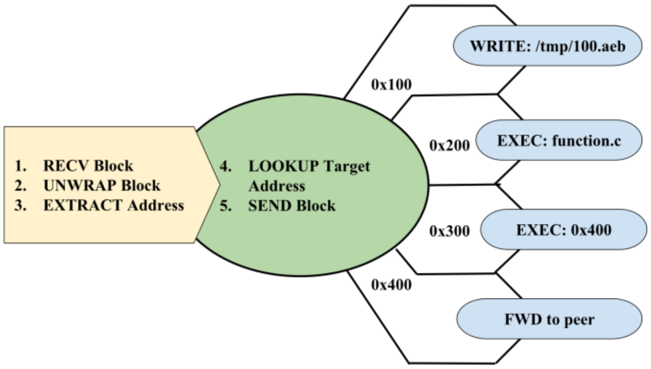
“Mesh” networking is a frequently deployed technology for managing both ad-hoc and planned networks. It is necessary, for a self-assembling network to operate with both resiliency and secrecy, to implement a data routing topology which is auto-learning, adaptive and heterogeneous.

A.6.1.1. Bootstrap

Current peer-to-peer systems use several methods to bootstrap new nodes to the network. Most rely either upon central directory servers, network flooding, or a pre-shared mapping file to begin the connection process. The AEB stores its network telemetry on the platform itself in the form of a Structured Document stored at a predetermined AEB Address. This metadata-containing Structured Document provide the bulk of information a new node needs to participate in the network, therefore the network connection and bootstrap process starts with the absolute minimum data and cryptographic keys necessary to decrypt and assemble these resources.

Furthermore, the controlled nature of AEB subnets, along with the encrypted Structured Document as a fundamental data structure, permit resource owners to either deploy their own secure bootstrap, or use a custom-factored confirmation process, such as two-factor authentication (2FA) to join the general network or controlled subnet.

A.6.2. Global Resource Map



Every Node Resource maintains a routing map, in the form of a Structured Document, which represents the entire available address space of the AEB universe. This Structured Document is segmented into address ranges, each of which is backed by a Function Map Application (FMApp). For every Block received by a Node, the destination address is extracted from the Block Headers and the appropriate FMApp is invoked.

The functionality of each FMApp is dependent upon its Key relationship with the Resource the FMApp function is covering. For most address ranges, a Block Routing is invoked to forward the block closer to the appropriate processing node. If the Block being routed is intended to be cached or stored, that FMApp can be executed as well. Ultimately, if the Node has appropriate permissions to the Resource, a custom FMApp application may be the final destination for the routed Block. The possibilities are endless.

A.6.3. Block Routing

The peer-to-peer AEB has properties of both hierarchical and mesh networking paradigms. Furthermore, the application of either paradigm's properties is configurable by a Structured Document Resource. This combination allows desirable features of both paradigms to operate between applications, depending upon the routing ruleset of a subnet or the interaction of rules between subnets. This creates block routing topologies which are heterogeneous, configurable, and extensible by application or plugin logic.

For example, if Alice's application requires data from Bob's subnet, Alice and Bob's subnets can be configured to peer directly with each other without intermediaries. Conversely, if policy requires that Bob's network telemetry stay anonymous, then routing rules can be deployed which obfuscate the source of Bob's data. Additionally, if policy requires Charlie keep copies of all of Bob's data for audit purposes, then rules which copy Alice-to-Bob block transfers also to Charlie can be deployed by Alice or Bob.

A.6.3.1. Topology Plugins

Fortunately, peer-to-peer mesh networking is well-studied and benefits from a multitude of existing protocol implementations and academic scholarship. There are literally dozens of these protocols, like Kademlia, IEEE 802.16, ZigBee, or Tor. Different protocols will continually be implemented and improved via topology plugins, and per-resource utilization of these plugins will be controlled by Resource Authority configuration Documents.

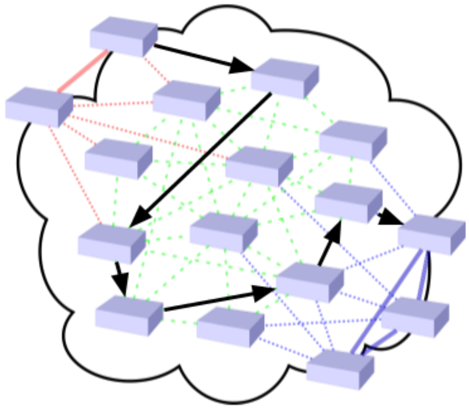


Fig B.2.2-1 Random Routing in a Mesh Network

A.6.3.2. Telemetry Sharing

Each node resource must manage its own aggregated communication topology. Towards this end, nodes will continuously, and at a configurable interval, provide telemetry information to all of it's peers, as well as relevant RA nodes.

From this feed, node resources will adjust route and topology configuration in a cycle of continuous testing and improvement, creating network resilience. Connections between resources will continuously improve speed and availability, and transparently support intermittent network scenarios and seamlessly recover from unplanned network partitions and other underlying connectivity issues.

Additionally, anomalous network errors, cryptographic key mismatches and other potential attack vectors will be stored and shared between peers and RA analysis resources for the purpose of identifying and mitigating malicious network activity. Ontological standards exist for sharing cyber threat intelligence data, such as STIX and TAXII, both by the OASIS group, and AEB telemetry data will be compliant with one or more of these standards, in order to facilitate threat and attack analysis for network and subnet owners.

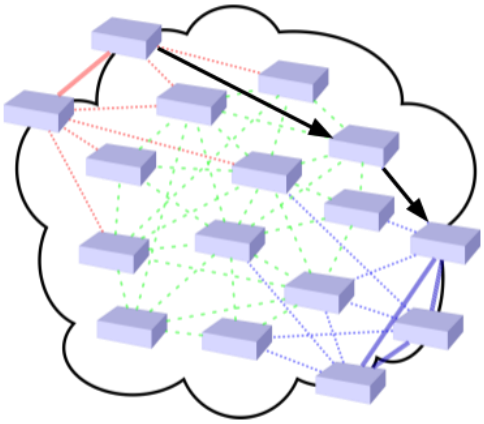


Fig B.2.2-2 Directed Routing in a Mesh Network

A.6.3.3. Peer Scoring

Telemetry data collected from peer resources will be continually aggregated into a matrix of score values. To allow multiple network topologies to coexist, an aggregated scoring metric is used to restrict different types of communications between nodes using radically different algorithms. This allows the network authority to disallow communications at high risk for errors or anomaly, and suggest alternate protocols.

A.6.3.4. Resource Mapping

All local data resource management functions will operate from a local Structured Document stored on disc or kept in memory. Since all AEB chain resources are composed of individual blocks, each encrypted separately with the continually evolving keys of the CDR, keeping this functional data in encrypted form when locally stored reduces the ability for an attacker with physical access to the hardware to decrypt and access the data. With custom hardware, RAM memory encryption, or disk-level encryption, attackers will suffer additional layers of cryptographic complexity deterring attack success.

Similar in function to IPv4’s 127.0.0.1 /8 CIDR block, all local operations, including application logic, will be conducted in a private block and mapped to external resources via a special subnet meta-chain. Even knowledge of other network resources can be kept from an attacker with common precautions, such as requiring a confirmation process to allow decryption.

A.6.4. Peer Transport

A primary feature of the AEB is resilience, that a node resource will take every step possible to transmit its message-block resources to its peers and ultimately, achieve consensus for its chain resources.

A.6.4.1. Transport Plugins

The act of transmitting a message between nodes is fundamentally concise and is already implemented by a multitude of protocols. As a heterogeneous platform where access control is managed cryptographically rather than by designated network port and protocol, the AEB operates block messaging with a plugin architecture allowing different and multiple transport types between nodes, and can operate under any existing messaging-based or similar environment, from public wifi hotspots, to foreign-controlled networks, to direct radio communications systems.

Additionally, transport plugins will support pre-existing protocols, such as HTTPS, SMTP, AQMP, or even third-party distributed systems like Tor, in addition to lower-level transports like raw TCP sockets, IPX Datagrams, or LTE or GSM wireless.

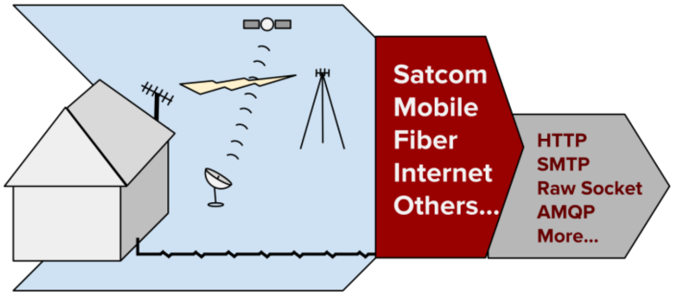


Fig A.5.3.1-1 Open Transport Plugin System / Multiple Options

A.6.4.2. Protocol Hopping

With a plugin architecture it is not necessary nor prudent for peers to communicate via only one transport mechanism. A node resource can peer as effectively via GSM wireless as it can via HTTP, so peer resource mappings must contain transport and connectivity credentials, allowing a node to peer via multiple transport mechanisms simultaneously. This obfuscates block transport over the network, since a node can communicate to one peer via HTTP on the open internet and to another via GSM wireless. Nodes operating on an intermittent or degraded basis can also continue to interact with remote resources.

A.6.4.3. Message Anonymizing

Since the AEB supports operation in environments with unknown levels of hostility, multiple techniques will be used in message-block transmission to deter attackers. In addition to ‘protocol hopping’, via configuration options, blocks can be nested in an “onion” formation, as in the Tor anonymity platform, reassembled in routing, as in the I2P platform, or other data formatting algorithms to increase difficulty for attackers.

A.6.4.4. Peer Chains

Peers will interact via their own private chain and accordingly, cryptographic double ratchet. This private chain will permit peers to disassemble and reassemble their inter-node blocks differently and distinctly from the resources the interaction represents.

A.7. Blocks

The Block is the smallest, lowest level, discrete structure of the AEB. It is simultaneously a message between two nodes or resources and the most discreet storage unit in the system.

A.7.1. Structure

A Block is structured in a binary format that consists of three primary parts, Flags, Headers, and Body:

A.7.1.1. Flags

The Flags field is a 64 bit unsigned integer separated into four (4) 16 bit sections:

- A Magic Number which identifies that the block complies with the AEB protocol
- A 16 bit section of bitwise binary flags that declare which unencrypted headers are expected to be present in the Headers field.
- A 16 bit integer ID that declares which built-in Key Derivation Function (KDF) function set is used, which built-in Cryptographic Hash Function (CHF) function set is used, which Encryption algorithm is used for Body content encryption, as well as designate the expected bit length of the included Public Key.
- A 16 bit integer reserved for use by the Applications or Resources which own the block.

A.7.1.2. Headers

The Headers field is a variable-length binary string comprising an appropriate value corresponding to each Header declared in the Flags field. The available Headers are, by name:

Target Address

The Target Address is one of two fields which must always be present in a block. As the name implies, this is analogous to a “To:” email header, and tells the AEB system how this block should be routed. This value may be an absolute Block address or a Resource address, as designated by flag values.

The difference between whether the Target is a Block Address or Resource Address is semantically the difference between whether you’re targeting a specific block within a Resource’s address range, or the genesis block of a Document Resource. This distinction becomes significant in Function Maps and related Structured Documents where the address is analogous to a function call.

Public Key

The Public Key is the other of two fields which must always be present in a block. This value is used by the AEB system to either lookup or create the appropriate Root Context for the content body corresponding to this key and Block Address. For common Double Ratchet operation, this key is used to create a new Diffie-Hellman shared secret which is the cryptographic basis for the corresponding Root Context.

Block Address

This optional Header designates the specific address of the Block within a Document Resource. Whether this Header is present depends upon the rules governed by the Flags field.

Parent Address

This optional Header declares which Block this given Block is directly a successor to. Somewhat analogous to the “From:” email header, this Header gives the system information necessary to both maintain the Chain Hash, as well as identify cryptographic transition points for Root and Block Contexts for out-of-order operations.

Content Hash

This optional Header contains the output of a CHF function as it is executed on the Body content of the Block. Depending upon Flag values, the value of this Header may also contain the Content Hash of the Parent Block, or the full Chain Hash up to the Parent Block. This is analogous to the “Merkle Root” hash of Bitcoin and other Bitcoin-based blockchain system.

Chain Hash

This optional Header contains the output of a CHF as it is executed on the Content Hash and/or Chain Hash of the Parent Block. Depending upon Flag values, the value of this Header may contain any of the Content Hash of the Parent Block, the calculated value of the Parent Block’s Chain Hash and Content Hash, or either calculated value and the current block’s Content Hash.

A.7.1.3. Encrypted Headers

Encrypted Headers are a special, optional section of the Block structure. If the appropriate bitwise flag is set in the Flags section above, a new section of binary data is created between the Headers and the Body of a Block. This section consists of a Flags integer and variable-length binary string comprising Headers in identical fashion to the standard Flags and Headers. However, this section is encrypted via an additional, optional cryptographic context child of the Root Context.

An Encrypted Headers section can contain the same Flags and Headers as listed above with some significant differences. Firstly, any Header, with the exception of the Target Address and Public Key can be overridden by a corresponding Encrypted Header. Secondly, the unencrypted Public Key header refers to the encryption context for the Encrypted Headers block rather than the Body, and an additional Public Key encrypted header is required to fulfill the body content Root Context purposes.

This Encrypted Headers methodology is directly adopted from the Double Ratchet Protocol’s optional Encrypted Headers protocol extension. It improves security by reducing the amount of open information available to potential adversaries, and does so by wrapping this information inside yet another encryption context, increasing cryptographic complexity.

A.7.1.4. Body

The Body section is a simple binary string, encrypted according to whichever algorithm is designated in the Flags section. The unencrypted format is totally open to suit the needs of the application controlling the resource, meaning existing applications can map its existing data structures onto Blocks and Documents based on those Blocks with minimal extensive rewriting.

A.8. Documents

Multiple Blocks chained together, within an Address Range, and sharing a Cryptographic Context, comprise a Document.

Documents are the core Application structure of the AEB. Whether a single Blockchain context Data Journal or a multi-dimensional Structured Document, Documents are what make the AEB unique and powerful. A Document can represent virtually any data structure from a simple binary blob GIF image to a complex database to executable code logic.

A.8.1. Journaled Data Documents

A Journaled Data Document (JDD) is the simplest extended (i.e. more than a Block) structure with the AEB ecosystem. It is represented as a defined Address Range as designated by an ownership key created by Resource Owner or Resource Authority. Functionally, it acts just like a common Blockchain and represents the equivalent of a single file or data set on a filesystem. It is Journaled in the sense that every change to the logical “file” is represented by one or more Blocks in chain sequence.

Example: Event Log

Many, many applications emit data in a forward-only log format, and this format very easily maps onto a simple Journaled Data Documents. Each log event, or collection of events, is added to a Document chain as a new Block.

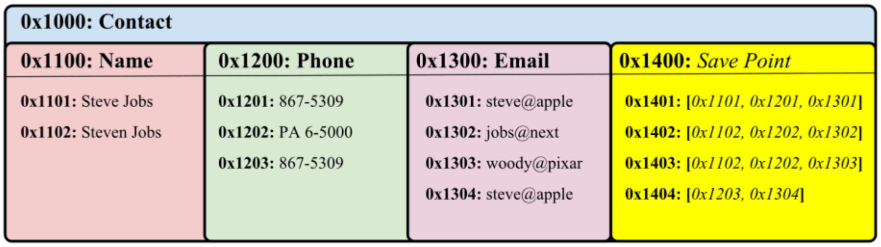
A.8.2. Structured Documents

A Structured Document is a segmented Address Range where each defined segment represents a data field, and each of these data fields can consist of a simple Journaled Data Document, or another nested Structured Document. It is with Structured Documents that a near-unlimited variety of data structures can be represented.

Example: Web Form

The most demonstrative use of a Structured Document is simple Web form-based application. For example, a landing page where a user submits their name, phone, and email address. To store this data we first need three fields: Name, Phone, and Email.

Each of these are deployed as Journaled Data child Documents within a Structured Document and three child address segments are allocated in the form of a map: “field



0x1000: Contact			
0x1100: Name	**0x1200: Phone**	**0x1300: Email**	**0x1400:** *Save Point*
0x1101: Steve Jobs	**0x1201:** 867-5309	**0x1301:** steve@apple	**0x1401:** [*0x1101, 0x1201, 0x1301*]
0x1102: Steven Jobs	**0x1202:** PA 6-5000	**0x1302:** jobs@next	**0x1402:** [*0x1102, 0x1202, 0x1302*]
	0x1203: 867-5309	**0x1303:** woody@pixar	**0x1403:** [*0x1102, 0x1202, 0x1303*]
		0x1304: steve@apple	**0x1404:** [*0x1203, 0x1304*]

A graphic representation of a Structured Document, with each child Journaled Document represented by an address range. Note how the Save Point field refers to the addresses comprising the “Save Point” action.

name" to Address segment. Whenever data is stored from the form's submission, a new Block containing the submitted data is created and appended to each field's child Document.

A.9. Function Map Applications (FMApps)

A Function Map Application or FMApp, is an extended type of Structured Document where one or more child addresses maps to executable code logic. Functionally, this means that addressing a Block to an FMApp address is the same as invoking a function within a program or an Application Programmer's Interface (API).

A.9.1. Beyond Smart Contracts

Beyond Smart Documents or Smart Contracts, a FMApp is the complete merger of AEB data assurance functionality with macro and micro scale application development. It is through FMApps that end-to-end data integrity is assured. It is the FMApp that gives the AEB a superior value proposition to other Blockchain-based platforms.

A.9.2. Embeddable Logic

An FMApp includes embeddable code, embeddable compiled binary, or reference to functionality built into the AEB daemon(s), in addition to the other structures embedded into a Structured Document, to perform its actions. These structures and functions may or may not be exposed to public use, may or may not contain wholly-private Address ranges for internal use, and may or may not make use of other FMApps to enhance its functionality. The whole purpose of an FMApp is to provide a concise service within the AEB ecosystem, with same cryptographic validity and assurance as the AEB itself.

A.9.3. Auto Distribution

Since an FMApp is a Structured Document, it's addressable anywhere on the global AEB ecosystem with the appropriate key-based permission. The Blocks of an FMApp can and will, depending upon application rules and key permissions, automatically and seamlessly distribute to whichever nodes need to invoke the code.

With this automatic distribution, applications can self-scale through configuration or additional logic and make use of additional computing Resources far beyond it's own scope. Imagine an IoT device or data collection drone in flight seamlessly, perhaps even secretly, engaging the 20 nearest computing nodes to process data from an important event captured on its sensors.

A.10.Developer's Interfaces

A.10.1. Application Programmers Interface (API)

The AEB core will include an API for developers to access platform structures and other resources, including but not limited to cryptographic functions like encryption and decryption, Resource Authority (RA) functions, accessing messaging and chain resources, and key creation, distribution, and validation.

Secondly, the API will express language environment libraries, including C, C++, Python, PHP, Java/JVM, Lua, Javascript, and others, for developers to make use of platform features and services in their own projects. Additionally, API functions will exist for developers to cryptographically sign their code to distribute and execute over the platform to their network.

APIs

Messaging
Scheduling / Cron
Plugins ABI
Storage
Service Proxy / VPN
Configuration
Encryption
Event Trigger
Routing Algorithm

A.10.2. Server API (SAPI)

A special type of API called a Server or Service API (SAPI), analogous to an Apache plugin, Java servlet, or NodeJS, will allow developers to build and deploy applications as services available both to other platform resources and to the public internet. Based upon the FMApp interface, developers used to coding for services and web-based applications will find a development paradigm highly familiar to traditional client-server and internet development. Also, encryption, transport, and distribution is handled entirely by the AEB platform, meaning developers and applications can take advantage of cryptographic assurance and security without needing any more than basic experience with secure development best-practices.

A.10.3. Services and Integral Applications

Certain applications and services, considered integral to the operation of the platform or strategically necessary for platform adoption will be included, as FMApps, managed by the Server API and expressed to developers by the API.

Sample of Services

Authorization & Authentication
Web Content Distribution (CDN)
Messaging (IM/Chat/Email)
Metrics & Network Analysis
Domain Name Service (DNS)
Social Svc (Forum/Blog)
Payments & Transactions
Web Svc (HTTP/SMTP/FTP)
Smart Documents

A.10.4. Virtual Machine

The Cryptographic assurance of embeddable code allows the creation of a technology that the computing public has not yet seen: a cryptographic virtual machine (CVM). A CVM is a virtual machine similar to the Java VM, the Erlang VM, or the WebAssembly VM, except that every input, output, and function of the CVM is cryptographically valid and even encrypted in-memory when necessary. This means that code written for the CVM can be cryptographically guaranteed to perform identically regardless of the environment the code is running under.

No more will the complexities of computing environment be significant, and applications will be cryptographically assured and secure even if they're executed in hostile, insecure, or mixed environments.

B. Use Cases and Errata

B.1. Intermittent Network Use Cases

B.1.1. Automatic Adaptive Peering

A full AEB node stores peer and routing data locally, using a FMApp. Upon loss of connection, it can be configured to attempt reconnection an arbitrary number of times. Since peers can operate over multiple transports, such as HTTP, Raw TCP socket, GSM datagrams, or other methods via plugin architecture, nodes have the ability to automatically adapt to it's own network environment and find at least one valid connection.

Also an AEB node can share peer connection credentials with other peers, depending upon cryptographic permissions, allowing "cellular"-style behavior to be configured for mobile nodes. For example, a mobile node, such as an Internet-of-Things (IoT) device, a mobile intelligence station, or even overhead satellite, can provide appropriate telemetry such as GPS coordinates, to a peer or subnet Resource Authority (RA), and securely receive connection credentials for the next available peer, maintaining a valid network connection even in intermittent or even hostile communications environment. Because the AEB is a nested hierarchy, an attempt at a Sybil attack (impersonating network nodes to subvert a consensus system) can be thwarted by referring back to the network's parent to verify nodes' identities.

B.1.2. Intermediary Block Distribution

In 'real world' operational network scenarios where connectivity between nodes is intermittent, as might be the case with mobile equipment or mobile teams, the AEB also supports intermediate block distribution (IBD). IBD can best be described as a form of 'baton passing', where a node resource, previously disconnected, transfers its data blocks to whichever cryptographically valid peer(s) it manages to connect with. This peer is then nominated and verified as a temporary proxy for the original node.

This asynchronous 'baton passing' is how block distribution systems work, even when connectivity is not in question, and is central for blockchain-based cryptocoin systems like Bitcoin to achieve network consensus. For intermittently connected nodes and subnets of nodes, block transfer of an entire team's transactions to the global network could occur via only one node. In a mobile team, for instance, all team members' blocks can end up being shared among all members of the team, and shared globally if only one team member manages to find a public internet hotspot or uplink.

B.1.3. Localization of Application Logic

A fundamental feature of distributed block systems is that an individual node has the ability to retrieve whatever resources it requires to perform its function from an upstream server (assuming it has access permissions).. The AEB platform, with its cryptographic assurance capabilities, also permits the localization of logic and code. Application software, cryptographically validated, can be seamlessly deployed and executed on remote nodes by encapsulating not only application logic (code) but also underlying data resources such as images or other media

For operational scenarios like Kiosks, Geographically remote stations, Smart Satellites, Mobile Teams, or Denied communications environments this means that entire application stacks are functional despite intermittent network connectivity, with the ability to bring up a redundant node to act as a proxy or substitute until comms can be reestablished.

B.2. Global Telemetry Intelligence

The global cryptographic assurance of the AEB results in anomalies, errors, and attack attempts to be easily highlighted and recorded. Combined with network telemetry from normal behavior, a great deal of information is available for analysis. These network events are logged to special logging blockchain resources and made available to peers and network owners. Additionally, if a network node also hosts clearnet Internet services, this data can be included for analysis and aggregation.

B.2.1. Distributed Signal Analysis

The distributed and segmented nature of the AEB, along with the ability to deploy cryptographically signed code for execution, creates a network, or subnet-wide, computing cluster which can act on collected data. Depending upon distributed code and configuration rules, node resources perform signal analysis: identification, verification, categorization, scoring, and if available, data correlation and enrichment from third party sources. This process is analogous to the map phase of the Hadoop distributed computer cluster software.

B.2.2. Aggregation and Application

Enriched anomalous signal data is good feedstock for a multitude of applications. Across a small AEB segment or globally, event aggregation can build a holistic view of attack patterns or identify systemic anomalies. This further processing, analogous to Hadoop's reduce phase, will provide applications and control systems with meaningful data for their respective decision trees.

B.3. Defenses Against Common Attack Types

Man-in-the-Middle Attacks
Certificate Authority (CA) features of Resource Authorities (RA) provide validation for encryption key resources. A source's RA establishes a secure 'side channel' blockchain to the destination's RA for validation purposes.

Man-on-the-Side Attacks, Forgery, and Fraudulent Credentials
Universal cryptographic validation and RA-to-RA validation (including RA of RA parent resource validation) stop MOTS attacks and provide valuable intelligence data.

Traffic Analysis / Correlation Attacks

Solved by “Onion Plus” block nesting and other block obfuscation methods including block splitting, block padding, in-transit partial reassembly, artificial timing delays in block transmission, and continual sharing of telemetry between peers.

Routing and other metadata-poisoning Denial of Service (DOS) Attacks

Universal cryptographic validation, including that of telemetry sharing, along with integration of routing and other metadata sharing algorithms into the platform core (vs. separate protocols as in the public internet) give these channels the same assurance as any other platform data resource chain.

Resistance to Cryptanalysis and Side Channel Attacks

Pluggable architecture for cryptographic primitives and logic, “Onion Plus” block nesting, and the continually evolving keys of the Double Ratchet mean that node resources are constantly operating with different keys and algorithms, exponentially increasing the difficulty for this type of attack.

Resistance to Sybil

Universal cryptographic validation utilizing hierarchical RAs by subnet, including key chain descent and credential replacement from a universal master key, silo these types of attacks into a deep-defense structure, allowing quick mitigation.

Injection Attacks such as SQL Injection

Under universal cryptographic validation, injection attacks are a form of credential forgery. Only those resources matching the correct address to crypto key have the authority, as validated by an RA, to write to chains.

C. Transition and Commercialization Strategy

C.1. Market Opportunity

C.1.1. State of Terror

Global Commerce is currently suffering a state of terror, with so many major data security breaches being made public on such a regular basis that news editors no longer give these events more than a blurb or a sound bite. However, according to the Ponemon Institute, in 2015 the cost of these attacks average USD $3.8 million per incident. Also, according to security firm Symantec's 2016 Internet Threat Report, there are nearly 35,000 'penetration events', i.e. attacks including partial and attempted breaches, per day. Furthermore, the British research firm Juniper Research warns by 2019 cybercrime will cost businesses and consumers over USD $2.1 trillion.

C.1.2. Dearth of Skill

The sheer quantity and impact of data security breaches globally is proof that application developers, by and large, are unable to build secure applications. With most underlying internet technologies designed for unencrypted, open communications, security is left entirely to application developers. Also, since modern application architecture is layered in nature, with libraries on top of libraries ad infinitum and written by third-parties with unknowable degrees of quality control, the probability that at least one developer in the stack carelessly introduced an exploitable vulnerability into your application is essentially guaranteed. The AEB, providing an underlying service resistant to attacks would grant protection without requiring every application developed to have a full security team doing perpetual audits.

C.1.3. Pervasive Requirement

Therefore, the market opportunity is to solve the first problem, the State of Terror, by proactively solving the second: secure application development, in this environment of incompetence, requires a platform tool chain that is equally as easy to build and deploy with as current development tools while providing a secure transaction infrastructure that is totally pervasive for end-to-end assurance yet is transparent for the journeyman application developer.

C.1.4. Leverage of Cryptographic Authority

The AEB, being rooted in global cryptographic key inheritance, allows global authority to provide favor to services it deems strategic. This ability to cryptographically designate 'default' services, and to embed cryptographically signed code logic directly into blockchain resources, will leverage the authority to generate income. For example, a default cryptocurrency and financial transaction service will generate revenue from transaction fees and currency demurrage. Additionally, standard API services such as Data

Storage, Instant Message and Email, Authorization and Authentication will provide additional revenue streams.

C.2. Market Identification

The AEB is not in itself a product. A product can be sold by any salesman to any target customer, just like any widget. The AEB is a foundational technology, and selling a technology requires the decision maker to be knowledgeable in the technology being sold and how it can be used to solve their business challenges. How a technology can be used to solve problems is a distinctly different value proposition than a technology that promises simply to solve a problem.

In most situations, the moneyed decision makers defer to their go-to “geek” which means the only market which genuinely matters for the AEB is developer mindshare and enthusiasm. If developers like the AEB and have easy tools to build with the AEB, they will build projects with the AEB, and they will recommend AEB-based solutions to their own customers and employers. Sendgrid, Wordpress, Amazon Web Services, MySQL Database, Github, Cloudflare, and others have all built their brands and market position on the backs of developer mindshare.

In short, the primary market for the AEB is Developers.

C.3. Top Challenge Points

The AEB is designed to literally replace several layers of the standard application stack, so just about any standard Internet-based product can be replicated on the platform. However, there are several challenge points which an AEB-based solution is best for:

C.3.1. Data Assurance with Data Security

Any application which can benefit from the Blockchain can benefit from the forensic data assurance of the AEB: Transactions, Asset Management, Smart Contracts, etc. However, where the AEB is genuinely needed, and inherently excels where the data itself must be secure and assured: Anything with Personally Identifiable Information (PII), Financial Data Compliance, or safety concerns sit at the top of that list. This includes:

- Health Care industry (HIPAA)
- Government Agencies subject to GSA Privacy rules
- PCI-DSS compliance

...as well as any environment where trade or government secrets face sustained attack from hostile actors.

C.3.2. Scale and Flexibility

The Function-as-a-Service capabilities of the AEB, as well as the flexible auto-deploy features of Function Map Apps (FMApps) make it the gold standard for scenarios which require flexibility and scale. This includes:

- Internet of Things (IoT)
- Data Collection and Analysis
- “Big Data” Processing

C.4. Memetic Market

Even though Ethereum, Ripple, and other Blockchain-based platforms currently have dominant buzz, the AEB platform is more closely aligned with standard software development paradigms dominant in Open Source, Enterprise, and the Tech Industry in general. Staking out a project, or business model, with defined goals and building a concise product to achieve those goals isn’t simply going to go away with a global ledger and a confusing array of “side chains.”

Buzz is a measure of potential. It is not a measure of market share. Containerization, Service Platforms, Feature Libraries, Cloud Computing, and the emergence of the Function-as-a-Service (FaaS) paradigm already have a much greater impact on developer focus and business priorities than coin platforms. This commoditization of platforms has been steadily marching forward since Amazon began AWS in 2006, and will continue until there exists a platform, like the AEB, which finalizes the process.

When the AEB emerges from “Stealth Mode” to offer the platform to the world, it will face fierce competition from both the Blockchain platforms as well as the Commoditization platforms. To combat this, Stratassure will enter the market space positioned as a revolutionary product fully prepared and with a “gamed out” plan to create and capitalize on first mover advantage. First, with the developers.

C.4.1. Developers

Among the paramount concerns Developers have when developing on or for a platform is whether the code involved is open and available. As such, the AEB platform daemon(s) will be available to download, compile, and build for anyone, under an Open Source license. Also, a non-profit foundation controlled by Stratassure will be created to maintain the core AEB project code, as well as maintain the global Resource Authority and all the global root keys for the AEB platform.

Having a non-profit operation control all-network maintenance functions gives public legitimacy to idea that the AEB is a serious platform for serious development. Also, a non-profit is an effective recruitment tool for developers as many top tier coders consider contributing to non-profit work ethically superior.

C.4.2. Astroturfing

"Astroturfing" is the political art of creating a movement and making it seem large and influential. Executed properly, no one is immune to this type of social engineering. To compete against the Commoditization companies like Amazon and Heroku, the AEB must build mindshare momentum by every means available, including rolling out Internet properties that can carry a "Powered by the AEB" label.

To accomplish this, Stratassure will publicly release several developer and consumer-friendly platform products in the months leading up to the full AEB roll-out. These products, not yet determined but likely to express primary API features integral to the AEB, will showcase the benefits of core AEB features, particularly evolving key encryption, automatic distribution, and data assurance.

For maximum impact, these products will compete directly with well known platforms and web properties, such as Sendgrid, Reddit, Twitter, Craigslist, Akamai, and/or Dropbox. Stratassure has identified no fewer than 20 major Internet properties which can be replaced with AEB-based competitors.

Incubating projects which use the AEB platform is a primary business goal for Stratassure, for income generation as well as for expanding the platform scope in both apparent and real terms.

C.4.3. Rapid Iteration

Rapid introduction of new features is a well understood method to build and maintain mindshare momentum among Developers, and Stratassure will make full use of this psy-op technique as part of the AEB roll-out program. The AEB will include at least 12 openly accessible Stratassure-written FMAapp-based APIs. Several core APIs will be released with the revelation of the AEB platform, with additional releases every few weeks, likely synchronized with related incubated product projects.

By maintaining product momentum and emphasizing Developer mindshare, Stratassure and the AEB platform, will establish memetic advantage over the much larger and less nimble incumbent market players. This is the exact strategy that enabled a very-young Sendgrid to not only survive the introduction of Amazon's S3 email product, but to thrive and still control it's target market of transactional email.

C.5. Maturation

As the crescendo of the "shock and awe" platform and market rollout, Stratassure will reveal itself as profit-seeking company with products to sell and support. It will sell three product lines:

C.5.1. Service Platform

Several APIs will be monetized in the style of a Function-as-a-Service platform, allowing developers to invoke Stratassure-developed functions on a pay-per-use basis.

C.5.2. Infrastructure Platform

Similar to Heroku, EngineYard, or Amazon Elastic Beanstalk, Stratassure will sell turn-key AEB-assured to developers and companies for their own AEB-based applications.

C.5.3. IoT/Mobile Toolkit

IoT devices often operation in extremely limited hardware environments. Stratassure will sell pre-compiled binaries for several hardware IoT platforms, including or excluding whichever AEB core features most common devices may or may not support.

C.6. Enterprise and Transition

In addition to the above 'retail' services, Stratassure Partners, or licensee, will provide enterprise network services, including secure network services and FMApp application development, to the DoD and other macro-scale government or enterprise needs. This will occur in partnership with major consulting firms and primary defense contractors, Business Development and Transition specialists, and drawing upon the Developer network built during the rollout phase for engineering resources,

C.6.1. Incubation

Product incubation will not stop with the end of the AEB platform rollout nor the establishment of the commercial Stratassure as a going concern. Drawing upon prior personal experience in the Home-based Business, Product white-labelling, and Small-Medium Business (SMB) markets, Stratassure will place a primary emphasis on building a global ecosystem.

With the AEB and accompanying APIs, an entrepreneurial journeyman coder in Warsaw or Manila is equally capable providing secure and assured data services to their local communities as any New York based mega contractor. Silicon Valley and the western tech industry in general almost completely ignores the rest of the world. Stratassure will partner with investor groups and incubator teams around the world to facilitate global adoption of the AEB for their secure data needs.

C.7. Timeframe

Stratassure expects to conduct the rollout describe above during the second half of Year 2, with the creation of Enterprise and Federal divisions during the Option Year 1.