

Keychain

Blockchain-based Public Key Infrastructure

System Description

Author: Jonathan Hope

1 Summary

Keychain is a system that enables secure messaging with end-to-end encryption and irrefutable digital signature attribution amongst known participants of the system. The system comprises of 1) a blockchain that stores and tracks linkages between encryption and digital-signature public keys on a distributed ledger and 2) supporting software and databases that enable participants to manage their public and private keys and interaction with other participants. The scope of this system covers the full range of data that may be transferred between two parties — from transient transactional messages to long lived documents. Throughout this document, we shall refer to this transferred data as messages for simplicity.

The blockchain is a distributed ledger that holds the public keys of participants. Participants may use the data on the blockchain to encrypt messages using the intended recipients' public keys and to verify digital signatures on messages that come from a sender. The encrypted (respectively, digitally signed) messages are sent to the recipients over independent peer-to-peer channels outside the blockchain (for example, point-to-point TCP/IP sockets over the Internet or an intranet relay server).

In scenarios where encryption or digital signatures are needed, a key risk-managing operation is replacing the currently used keys with newly generated keys. This operation serves as a best-practice measure against brute force attacks by hackers to reverse engineer the private keys as well as a recovery step following a break-glass event or a security breach. Keychain provides the infrastructure to allow participants to communicate key-change events to their known counter parties, providing authentication proofs, while allowing each party to maintain pseudonymity with respect to parties that are not related to the specific communication channel, even if they are participants in the blockchain system.

The Keychain system performs key update, notification, and validation automatically after the first public key data is exchanged by the relevant parties. Once parties have set up the public keys on the blockchain and have established peer-to-peer communications, the encryption, decryption, signing, and signature verification are performed by the Keychain system transparently – allowing Keychain to be used transparently in conjunction with existing message protocols that operate over the Internet Protocol stack such as the FIX, FAST, the Japanese Zengin, or HTTPS protocols. The Keychain system is data agnostic and applies in situations where messages, raw data, or documents are being transferred and/or stored between one or more parties.

Use cases include, but are not limited to the following:

1. Secure financial messaging
 - A) Central banks and member banks that exchange in real time business-critical financial transaction messages will use the Keychain as an additional authentication factor and end-to-end encryption tool to ensure message integrity over disparate networks.
2. Secure email
 - A) An online merchant will use Keychain to digitally sign emails so that their customer which use the Keychain may automatically verify the origin of the message
 - B) A credit card issuer may want their customers to digitally sign emails that request account modification to reduce incidences of fraudulent requests from hackers.
3. Financial securitization and financing
 - A) Commercial banks and insurance companies will use Keychain to validate digital signatures by counter parties on commercial bills and electronic records of monetary claims to under write financing and policies

See Fig. 6. “Process-driven Communication Configuration” and Fig. 7. “Secure Email Communication Configuration” in the “Keychain – Patent Diagrams” document for sample contexts of applications of the Keychain system.

2 System Description

2.1 System Elements and Definitions

2.1.1 Blockchain

Keychain is a network of Keychain nodes (applications) run by participating parties that replicate and store the Keychain keys in a cryptographically linked data store known as the blockchain. An example of a similar, but unrelated blockchain system is Bitcoin. Keychain’s blockchain architecture is similar to and based on that of Bitcoin, hence this document will not cover the basic design of the blockchain. See Fig. 2. “Anatomy of a Keychain” in the “Keychain – Patent Diagrams” document. Unlike Bitcoin, however, Keychain can function as either a public blockchain or a private blockchain.

2.1.2 Transactions

The Keychain blockchain records state changes as a series of transactions and blocks, and new transactions refer to previous transactions so that the nodes on the network are able to validate the new transaction. We call a transaction a descendant of another transaction if the former’s input is the output of the latter.

Transactions in Keychain store payload data that includes a public key and metadata describing the public key using the OP_RETURN script code. The payload data is formatted using an extensible, self-describing data format such as XML or JSON so that the system may be easily modified to support new types of public keys in the future. Such changes in the system will be soft network forks as opposed to hard forks.

Keychain differs from Bitcoin in that new transactions created by an end user have a single output to an address that was also newly created by the same user. The user which controlled the private keys of the previous transaction also control the private keys of the new transaction. Such transactions carry a different payload, however control is maintained by the same user thus creating a link between the different payloads. Participants use this aspect to deduce that the new payload represents a replacement of the older and that both came from the same end user. Given this, participants may trust that off-chain messages that are encrypted or signed by the new public key come from the same source as the previous public key, and this forms the basis of a communication pattern where authentication outside the blockchain is anchored by transactions in the blockchain.

A sample transaction in human-readable ASCII format is the following:

```
{
  "hex":
    "010000000101a6f1fd1136c8cdd056043cec88740b70664b6b644433711b26e92bdb54d4c00000000494
    83045022100eff964d23fd40ae50a1787e6642b5dfdac7a853f357676208909c2c3f45d6ef5022012dcaef
    c25d2a4bba1fba76a3c6db09d679e29f4402d86ccdd66c03286257b3a01ffffffff02f0ca052a010000001
    976a9140e5e8f18b99f56e2ff104398f4c9b28a1e0d5fdc88ac000000000000000fd88026a4d84027b226
    66f726d6174223a202243727970746f70702d352e362e322d44455222c202276657273696f6e223a20224
    34d502d302e312e30222c202274797065223a2022656e6372797074696f6e222c20226b6579223a2022141
    414143082020c02820101008f2de41d58e8b4107a2e53fa4b99659ac2b3db6d55d5c38d73e1e27bd3909d3
    fc1388cef5d1b92176c66133c63112d4e68c96c450ee62cbae20d5fb549fc496efbe740204b7ae7265399a
    841885bb0baa54d5aae5e640a09e1c30957c71344b0ccce5055e332db55850e9369516525b19ebb75cf45c
    3dca01cc137077973eedd4e3630eb06f90410077e1aeeab7ea6882cbb0dbb39042d108138b48a2673e27b
    3b24d38bfb58dcf4987fa097e55eac280b7b5493fd4a458bc640f7297e435c0d6e6b06aad0acf48cdb77fe
    b4ac7fc83d18cd053a33e449c24edc49c67a791327f032468d91bf2b007c2086316984daaedbe174da0db2
    a2cfa007886d69a8a67020102028201003baca96ae05106e459f3b8589d8e5c99724b8c43f5a2f08f35273
    3771de0fd8980b6b4756637178a52fa3e150cbde5a548b996352573bb8e3437fb1a2a4633dfff83b0ce983
    079d85c366cbc7a6ac676fb7534f4ad3a1dffb967909e6995495c5e7f4647bb3de2a087941805613c9b240
    9c7f5e0b5d4e5ac3f35ef15124ed87022ec50e17a16da26628f16d0a16529f9a797939ad56c390845d520a
    2a17824e86d7e07fb4ddc53ff6bcf120f6dfc2e339e77d178478873f06eb9ba4ec4d8d547ef57347804ed6
    1ac35d9b11c1f7c0bd92c4abb1ea6371385a9fc119066bcd61ed989d5cbc0a5726183fc7b04a9806d92126d
    cd5bd8651243e637eb3f549962c7714141414222c202264657363223a2022456c47616d616c227d0000000
    0",
  "txid": "10e062107442df82f7316198a0ff1f6a3b1d2655a4e6076bcbcf4fee6c8a6763",
  "size": 817,
  "version": 1,
  "locktime": 0,
  "vin": [
    {
      "txid": "4c4db5bd926eb211373344646b4b66700b7488ec3c0456d0cdc83611fdf1a601",
      "vout": 0,
      "scriptSig": {
        "asm":
          "3045022100eff964d23fd40ae50a1787e6642b5dfdac7a853f357676208909c2c3f45d6ef5022012dcaef
          c25d2a4bba1fba76a3c6db09d679e29f4402d86ccdd66c03286257b3a[ALL]",

```

```

      "hex":
"483045022100eff964d23fd40ae50a1787e6642b5dfdac7a853f357676208909c2c3f45d6ef5022012dca
efc25d2a4bba1fba76a3c6db09d679e29f4402d86ccdd66c03286257b3a01"

    },

    "sequence": 4294967295

  }

],

"vout": [

  {

    "value": 49.99990000,

    "n": 0,

    "scriptPubKey": {

      "asm":      "OP_DUP      OP_HASH160      0e5e8f18b99f56e2ff104398f4c9b28a1e0d5fdc
OP_EQUALVERIFY OP_CHECKSIG",

      "hex": "76a9140e5e8f18b99f56e2ff104398f4c9b28a1e0d5fdc88ac",

      "reqSigs": 1,

      "type": "pubkeyhash",

      "addresses": [

        "mgpw2p4Guj5i3VdhLtArEzWK6c7V89oKQf"

      ]

    }

  },

  {

    "value": 0.00000000,

    "n": 1,

    "scriptPubKey": {

      "asm":
"OP_RETURN
7b22666f726d6174223a202243727970746f70702d352e362e322d444552222c202276657273696f6e223a
2022434d502d302e312e30222c202274797065223a2022656e6372797074696f6e222c20226b6579223a20
22141414143082020c02820101008f2de41d58e8b4107a2e53fa4b99659ac2b3db6d55d5c38d73e1e27bd3
909d3fc1388cef5d1b92176c66133c63112d4e68c96c450ee62cbae20d5fb549fc496efbe740204b7ae726
5399a841885bb0baa54d5aae5e640a09e1c30957c71344b0ccce5055e332db55850e9369516525b19ebb75
cf45c3dcda01cc137077973eedd4e3630eb06f90410077e1aeeab7ea6882cbb0dbb39042d108138b48a267
3e27b3b24d38bf58dcf4987fa097e55eac280b7b5493fd4a458bc640f7297e435c0d6e6b06aad0acf48cd
b77feb4ac7fc83d18cd053a33e449c24edc49c67a791327f032468d91bf2b007c2086316984daaedbe174d
a0db2a2cf007886d69a8a67020102028201003baca96ae05106e459f3b8589d8e5c99724b8c43f5a2f08f
352733771de0fd8980bb6b4756637178a52fa3e150cbde5a548b996352573bb8e3437fb1a2a4633dfff83b0
ce983079d85c366cbc7a6ac676fb7534f4ad3a1dffb967909e6995495c5e7f4647bb3de2a087941805613c
9b2409c7f5e0b5d4e5ac3f35ef15124ed87022ec50e17a16da26628f16d0a16529f9a797939ad56c390845

```

```

d520a2a17824e86d7e07fb4ddc53ff6bcf120f6dfc2e339e77d178478873f06eb9ba4ec4d8d547ef573478
04ed61ac35d9b11c1f7cbd92c4abb1ea6371385a9fc119066bcd61ed989d5cbc0a5726183fc7b04a9806d9
2126dcd5bd8651243e637eb3f549962c7714141414222c202264657363223a2022456c47616d616c227d",

    "hex":
    "6a4d84027b22666f726d6174223a202243727970746f70702d352e362e322d444552222c2022766572736
96f6e223a2022434d502d302e312e30222c202274797065223a2022656e6372797074696f6e222c20226b6
579223a2022141414143082020c02820101008f2de41d58e8b4107a2e53fa4b99659ac2b3db6d55d5c38d7
3e1e27bd3909d3fc1388cef5d1b92176c66133c63112d4e68c96c450ee62bae20d5fb549fc496efbe7402
04b7ae7265399a841885bb0baa54d5aae5e640a09e1c30957c71344b0ccc5055e332db55850e936951652
5b19ebb75cf45c3dcd01cc137077973eedd4e3630eb06f90410077e1aeeab7ea6882cbb0dbb39042d1081
38b48a2673e27b3b24d38bfb58dcf4987fa097e55eac280b7b5493fd4a458bc640f7297e435c0d6e6b06aa
d0acf48cdb77feb4ac7fc83d18cd053a33e449c24edc49c67a791327f032468d91bf2b007c2086316984da
aedbe174da0db2a2cfa007886d69a8a67020102028201003baca96ae05106e459f3b8589d8e5c99724b8c4
3f5a2f08f352733771de0fd8980b6b4756637178a52fa3e150cbde5a548b996352573bb8e3437fb1a2a463
3dfff83b0ce983079d85c366cbc7a6ac676fb7534f4ad3a1dfbf967909e6995495c5e7f4647bb3de2a0879
41805613c9b2409c7f5e0b5d4e5ac3f35ef15124ed87022ec50e17a16da26628f16d0a16529f9a797939ad
56c390845d520a2a17824e86d7e07fb4ddc53ff6bcf120f6dfc2e339e77d178478873f06eb9ba4ec4d8d54
7ef57347804ed61ac35d9b11c1f7cbd92c4abb1ea6371385a9fc119066bcd61ed989d5cbc0a5726183fc7b
04a9806d92126dcd5bd8651243e637eb3f549962c7714141414222c202264657363223a2022456c47616d6
16c227d",

    "type": "nulldata"

}

},

    "bLockhash": "067dc452bde847b5dde19bf8b7be83069fbb098b4a338f4de5573d06477b9e84",

    "confirmations": 11,

    "time": 1461295241,

    "bLocktime": 1461295241

}

```

2.1.3 Keys

The Keychain maintains two classes of keys. The first class is used, as in Bitcoin, for pseudonymous authentication and verification of transactions on the blockchain. Like Bitcoin, Keychain uses Elliptic Curve Digital Signature Algorithm signature keys for this purpose.

The second class of keys, which does not exist in Bitcoin, includes application-level public keys that are used for encrypting/decrypting and/or signing/verifying data outside of the Keychain blockchain structure. This second class has no relevance to the maintenance of the blockchain itself, thus has purpose independent of the first class. This class comprises of two types of public keys: encryption and digital signature keys. As alluded to earlier, the primary purpose of the Keychain is to record and track transaction linkages between keys of this class on the blockchain.

Keychain application-level public keys are embedded in a transaction, and the end users may choose at their discretion which encryption or digital signature keys are appropriate for their purposes. The set of supported key types include but are not limited to the following:

1. Encryption
 - A) RSA
 - B) El Gamal
 - C) Elliptic Curve Integrated Encryption Scheme
2. Signatures
 - A) Digital Signature Algorithm
 - B) Elliptic Curve Digital Signature Algorithm

Message recipients are required to support these key types and algorithms, and these are supported in the Keychain reference client.

2.1.4 Keychains

A keychain (the lower case 'k' is significant) is a sequence of application-level public keys and associated meta-data that are the payload of linked blockchain transactions. In this document, we use the term keychain (with a lower-case 'k') to refer to the above sequence of keys while we use Keychain (with an upper-case 'K') to refer to the entire system or the blockchain, whichever is most appropriate in context.

The keys within a keychain form the basis of authentication and represent a unique contribution of the Keychain system. The first key in the sequence is called the *root*, and the last is called the *tip*. A key A is called a *descendant* of a key B if and only if A's transaction is a descendant of B's transaction.

The meta-data contains the following information:

1. Key type - encryption or signature
2. Key description – for example, El Gamal
3. Key format – which crypto implementation or format (for example, 'crypto++ DER format')
4. Version – implementation version

Meta data have no personally-identifying information. Rather, participants (who are independently informed of the identity) use this inductive logic: if a new transaction extends a previously authenticated keychain, the new key shall be considered as being controlled by the entity that controlled the previous keychain and therefore trusted. Each participant that extends a keychain on the blockchain are responsible for ensuring the following:

1. The participant has secured the private keys for both the blockchain-level and application-level public keys so that no unauthorized party can independently extend a keychain of the participant or impersonate the participant in the peer-to-peer communication channel with other parties.
2. After creating a new key pair internally, the keychain is extended with the new application-level public key that corresponds with the new private key.
3. The application-level key's type and format is support by the intended recipients.

A participant may have multiple keychains on the blockchain for multiple recipients. The Keychain software tools implement the above logic seamlessly on behalf of the participant.

2.2 System Components

Each participant runs the following components:

1. **Keychain daemon full node** – a process that validates transactions, mines blocks, and persists the blocks to a database.
2. **Keychain Keyring** – an application that generates new application-level and blockchain-level key pairs, persists the private keys to a database, and constructs blockchain transactions on behalf of the end user.
3. **Keychain Gateway** – a messaging gateway that relays messages to and from counter parties, encrypting/decrypting/signing/verifying the messages as desired by the end user. The gateway is available as an application and a library.

Applications of the Keychain system may have other components such as application plugins to integrate the Keychain functionality into other applications such as Microsoft Office. See Fig. 3. “Keychain System Components” in the “Keychain – Patent Diagrams” document.

2.3 Usage Workflow

Messages may be sent between senders’ and recipients’ gateways in one of three transformations:

1. Encrypted
2. Digitally signed
3. Digitally signed after being encrypted

2.3.1 Encryption

Assuming the root encryption public key has been shared by the intended message recipients with the sender, the sender’s gateway process does the following to encrypt the message:

1. For each recipient, retrieve the tip of recipient’s keychain from the Keychain blockchain.
2. Encrypt the message using authenticated, hybrid encryption.

Authenticated, hybrid encryption is a well-known technique where the message is encrypted using a symmetric cipher such as the Advanced Encryption Standard (AES) with Message Authentication Codes (MAC), the symmetric key is encrypted for each public key using asymmetric public-key encryption (such as RSA), and the both the encrypted message and a list of encrypted symmetric keys are sent to all the recipients.

Let m be a plain text message. Let (E_1, D_1) be a secure symmetric encryption scheme with MAC authentication of associated data, such as AES-128 in GCM mode. Let (E_2, D_2) be a secure public-key encryption scheme (such as RSA or El Gamal), and (pk_i, sk_i) be the set of public and private keys for each recipient. To broadcast the message, the sender randomly chooses a symmetric key k and sends

$$E_1(k, m) \parallel pk_i \parallel E_2(pk_i, k) \parallel pk_2 \parallel E_2(pk_2, k) \parallel \dots \parallel pk_N \parallel E_2(pk_N, k)$$

to all of the recipients, where $\parallel$ denotes concatenation.

The use of hybrid encryption is a key aspect of the utility of an authentication system like Keychain in performing more complex messaging patterns such as *authenticated Byzantine Agreement*. See Fig. 4. “Encryption and Decryption” in the “Keychain – Patent Diagrams” document.

2.3.2 Decryption

Assuming the root encryption public key has been shared by the intended message recipients with the sender, the receiver’s gateway process does the following to decrypt the message:

1. Search the message’s list of public keys for the recipient’s keychain tip.
2. If present, decrypt the corresponding encrypted symmetric key with the private key corresponding to the matched public key.
3. Decrypt the encrypted message with the recovered symmetric key.

Let m be a plain text message. Let (E_1, D_1) be a secure symmetric encryption scheme with MAC authentication of associated data, such as AES-128 in GCM mode. Let (E_2, D_2) be a secure public-key encryption scheme (such as RSA or El Gamal), and (pk_i, sk_i) be the set of public and private keys for each recipient. To decrypt a received encrypted message $E_1(k, m) \parallel pk_i \parallel E_2(pk_i, k) \parallel pk_2 \parallel E_2(pk_2, k) \parallel \dots \parallel pk_N \parallel E_2(pk_N, k)$, the recipient finds its public key in the cypher-text, and computes

$$m = D_1(D_2(sk_i, E_2(pk_i, k)), E_1(k, m))$$

with its corresponding private key. See Fig. 4. “Encryption and Decryption” in the “Keychain – Patent Diagrams” document.

2.3.3 Digital signature

Assuming the root signature public key has been shared by the sender with the intended recipients, the sender’s gateway process does the following to sign the message:

1. Sign the message with the private key corresponding to the public signature key that is the tip of the sender’s keychain.
2. Send to the recipients the message, the signature, and the public key.

Let m be a plain text message. Let (S, V) be a secure digital signature scheme. The sender sends

$$m \parallel S(sk, m) \parallel pk$$

where pk and sk are the public and private keys. See Fig. 5. “Signing and Verification” in the “Keychain – Patent Diagrams” document.

2.3.4 Signature verification

Assuming the root signature public key has been shared by the sender with the intended recipients, the recipients’ gateway processes do the following to verify the message’s signature:

1. Confirm whether the public signature key included in the message is on the tip of the sender’s keychain.
2. Verify that the signature is valid for the message.

Let m be a plain text message. Let (S, V) be a secure digital signature scheme. The recipient confirms $V(pk, m, S(sk, m)) = 1$, where pk and sk are the public and private key of the sender's keychain tip. See Fig. 5. "Signing and Verification" in the "Keychain – Patent Diagrams" document.

2.3.5 Message Format

Messages transformations are rendered in XML format, separating the components of the packed cryptographic data into a self-describing, extensible, ASCII format. Binary data is hex encoded.

An encryption of a sample message "Got it!!!" with AES-128 and El Gammal is the following:

```
<SecureMessage Timestamp="2016-05-08 00:57:24.977105+00" Cipher="AES" PKI="ELGamal">

  <AssociatedData>

    <InitialValue>320DD6A217FB90F9C614CE7DCC042253</InitialValue>

    <EncryptedSymmetricKeys>

      <EncryptedSymmetricKey>

        <CipherText>099565EBCB42848A8DE7C64E12E06789917A05E889D00DBD19F9433663C8EA87B5AF40D9A2
        B1D6B59522AF33CA4D925EE9C60357C70DE27A32510DD082CD762A206B18DD2EADF97C4914312ECC69C570
        1225950809C31D30D2BF1F8AD33AE584B4C1A66960DF8CB685CB06CD96B519D7245CD1871F3CC2EF504115
        BBC74FCB75C42E02CED0154438A9A4AC22CD8FA96221AB2B92893675F959AC5F79F01A765C6F45AEFD7B33
        FB0D18EB9F0A89AC666E5304DDD7173B0B47937E303BB35A3810B9628C5E9CD7D2EB8340D1892461217E67
        F5439FF0DC723F4ADFA8DCE994F27BC10EB211DE05356506FEAFF4BE7CC5B156AD66AC3510D1B3B8D4067
        56B661FB912D2FF1E4F561B46A78D3115516FEBA90D737865A2B134443DE398303824DC9E7F774B82DE6C2
        044BF76D561BA21A080A5DBB6532327E78C67912C10BA563114E3C757665FEA767A8D82518FD67AA168F57
        A541042B9445FE026945135B941766AE8537CAA9CAF44606892566789CE1A63BEB4A4D4297432EEAD1F509
        B5E70EF0E7EBFD46DCAED633FA1B7D5F15B20753DA1E3BC2C2C51A1BE9AA85BD1E670A760EBFC92ACF796F
        378803D8057DEB21B3EE878BC6FFAB074B98D3F72F85794F47BB2D13608F4FF3D39455AA534C4091B90581
        AB8434C175AF720E5399D215DA6BA1320A5779F8E7842E323CEAFF3DC38800E65404457684FFB742E7947F
        CDC3</CipherText>

        <PublicKey>3082020C02820101009670B338BB41C8226DBE56318ABA75BC69176801BC2D50D6D27D7BF05
        618CCBDCFED591AAC8D4831884441B3A7C488E45C087B033E8AA2DB49B128495C88E49B3741696CA9F2B38
        41A7E8F4D5E5937A6C568FFE772356C38288011830F07257FC1D39DDA3BE2961F7EAF55118CF8AADB13CDD
        23296CB3391BB3EC0ACD5D025320AC2A39FB546D28A0D13C56CE8B8964A86D15B02153B01445217E7A9040
        8991320E5E6F0A1DF7579D0A18FE43168484A162352707F65A5ABC2F35B7E61A9358E0E9B7506E8DF12D35
        64A747FDC9586DB3DDE85D161715B56207EF30896E1728B6F7DA325C4B293607A9C4EA9C394A9D6B669CA4
        1E552FE5D65FCA6E07DC9CD4B0201030282010068F6723C7A02DE23B2E508A3F58AD4FFDF7D39A83B91D28
        9DAD73AEFE3FD899F760F59708DA6D7A07E1B92B2D78CA77C6E9A21425FDC0EA59BDD6409130716A2ABE79
        2565FEC8B2AC11461CD7E01144794A887FF58BAFE638750A6ACC9613075FD63BB971658A9086C89397A857
        B3516E12A859B6601FE78B9F646341839E4183AE7AD5A32F40E1E273C5E67F0F741DB095C08EA7930A4C3A
        5FE5D5B23D20B5089ED9E5A52AC0AFAD3AD079ECC95CAB93C6837EE1056B6F12351B50EF7CBDAE9F920714
        5FBECAC4AD095878117001AE23D201F478CC93D21DB933D4B3B801080D7A3557613FD57D4EE0C2206A9C68
        0B7ADFA7303A5D4483F51B4193BF466ABEC</PublicKey>

      </EncryptedSymmetricKey>

    </EncryptedSymmetricKeys>

  </AssociatedData>

  <EncryptedMessage>9ECF4DE2D0F6409F2F604409EB0A7A427AC92B687DF0E01602</EncryptedMessage
  >
```

```
</SecureMessage>
```

Note that XML tag names are subject to change for later versions of the software. See Fig. 9. “Encryption and Decryption of Sample Message”, Fig. 10. “Signing and Verification of Sample Message”, and Fig. 11. “Signing and Encryption Composition of Sample Message” in the “Keychain – Patent Diagrams” document for more samples of transformed messages.

2.3.6 Keychain extension synchronization

As the propagation of blockchain state is asynchronous to the peer-to-peer communication channels, race conditions exist for whether a sender and recipient encrypt/decrypt or sign/verify using the same keychain tip, despite them having already exchanged the root key. Keychain mitigates race scenarios by using the block confirmation state as a synchronization anchor. The use of the block confirmation state as a synchronization anchor for off-chain communication is another key contribution of the Keychain design.

Suppose a participant extends his encryption keychain and suppose that the transaction is mined into a block. While all participants receive notification of the extension upon the mining of that block, a sending participant will only begin to begin to encrypt with the new public key after K confirmations are received, and the participant will begin accepting messages encrypted with the new public key after only $K/2$ confirmations are received. The same rule applies to the signature/verification case. The rule is enforced by the Keychain gateways based on session configuration for K . A reasonable number for K is six.

2.3.7 Initial key exchange

Participating counterparties exchange the root key independently of the Keychain system and can be done in any manner that suits the parties for the given situation. In situations where the data to be sent is strictly confidential with a high-value loss potential in the event of a security breach, it is recommended that the parties exchange the root key in person. In casual situations, sending the root key over the Internet might suffice. Note that once the root key is exchanged, all key changes are handled automatically by the Keychain system.

When root keys are exchanged, parties pass a QR image encoding of the blockchain transaction ID and/or blockchain address of the transaction that holds the root key. See Fig. 8. “QR Code of a Sample Root Key Transaction ID” in the “Keychain – Patent Diagrams” document for an example QR code. The parties import the image into the Keychain Keyring application, which then queries the blockchain to extract the root key. Root keys can be shared only after the transaction carrying the root key has been mined.

Keychain’s design was created with the recognition that in-person key exchange is a perfectly valid step in a secure system, given that for example every financial institution initiates electronic connectivity services with in-person contract negotiations and client onboarding. The initial Keychain root key exchange is intended to fit seamlessly into this well-established business practice amongst the financial industry.

The key to this utility patent application is the belief among the creators that Keychain’s business-centric design, informed by years of experience in the financial industry, defines a valuable

innovation not only for what the solution provides but the simplicity allowed by not solving irrelevant problems. Thus the omission of any extraneous technical framework to protect the initial key exchange is by design and should be considered as part of the innovation of this system.

2.3.8 Transaction funding

Keychain has a transaction accounting and proof of work system similar to that of Bitcoin. Each transaction carries a value amount and fees are paid to miners for mining blocks. End users may fund their keychain extensions by adding non-keychain transactions as an input to the keychain-extending transaction. This is managed automatically by the Keychain Keyring application.

3 Differentiators

1. Keychain is a public-key infrastructure system, not a distributed asset ledger.
 - A) The purpose of Keychain transactions is to signal a key change to all interested parties without revealing any information about the content of the parties' peer-to-peer communication.
 - B) Keychain does not store application-level transaction data on the blockchain
2. Keychain does not store any identifying data.
 - A) The participants maintain identifying information such as keychain labels locally with the help of the Keychain Keyring application.
3. Keychain is scalable.
 - A) Participants can exchange a practically unlimited number of messages that are secured by a single keychain transaction.
 - B) The unofficial observed latency for the inline encryption/decryption/signing/verification steps is in the low hundreds of milliseconds.
 - C) Keychain data is cached by the Keychain Gateway so that inline communication avoids the latency of querying the daemon.
 - D) With multithreading, the Keychain Gateway can process several thousands of messages per second on hardware of sufficient caliber.
4. Keychain integrates easily into existing IT infrastructures.
 - A) Keychain provides transparent end-to-end communication so that existing communicating processes may operate oblivious to the fact that the communication is secured by Keychain.
5. Keychain can function as a private, permissioned blockchain as well as a public blockchain.
 - A) As Keychain is a mechanism for authenticating participants, Keychain can uniquely authenticate itself by having a widely publicized signature keychain on the blockchain. When other nodes connect to the network, the Keychain node sends out a list of IPs from which connections are allowed. The other nodes use this list to allow or block incoming connection requests.