QwyitKey™ Overview

This document provides an overview of the QwyitKey™ *Anonymous Authentication Service* (QwyitKey™), the world's first participant-managed, independent-trust authentication service for secure messaging. QwyitKey™ is an implementation capability based on the full Qwyit protocol as outlined in the current version of the *Qwyit Protocol Reference* document, available from Qwyit LLC. QwyitKey™ client demo APIs are available from Qwyit LLC; go to www.qwyit.com.

Introduction

As we've pointed out through Qwyit's history, there is a fundamental problem with security technologies: they aren't used in every communication, in every digital application, they aren't baked into the multiple device architectures across all the various digital networks. If they were, almost all of the problems – from network intrusions and data theft, to financial fraud – would be, if not solved, severely limited. The digital world would mirror the physical one: crime rates below 5%, instead of the unbelievable attacks reported every day.

This is just a common truth:

If there was universal, end-to-end network security, there would be minimal criminal activity, limited to inside jobs and local attacks: just like the real world.

[When the digital Equifax break-in gave out 140+*MILLION* records, this would be the same in the real world as if every single bank branch in the entire US was *successfully* broken into at the same time. Digital Security is a complete failure.]

Without mutual, constant authenticated and end-to-end encrypted communication, it's easy to 'find your way around' – and then do whatever you want, wherever you are, including Digital Bad Things. But if *everything* – every digital communication – looked like a foreign language, security is the default. Crime becomes something incredibly difficult, with obvious moral failings; not something anyone can do without even trying.

Again, as we've pointed out previously, what's the answer to why *everything* isn't secured? There are two problems, one glaringly apparent and the other quite subtle:

- There's no universal, simple, fits-everywhere method
  - This is because the current methods aren't able to be universal – Qwyit has pointed out the defects for years now: too big, too complex, too slow, too…much
- Networks have become multi-dimensional, making end-to-end authentication and encryption impossible using current methods

Universal Security

A summary example of the inability of current authentication and encryption methods to proliferate universally: The latest version of TLS (V1.3, August 2018), is touted as 'Faster and More Secure' than previous versions (because, these are the two things that previous versions *weren't!*) Yet, the 'Faster' is because there is no longer a need to authenticate even once per session (which is the problem with

most network intrusions: nobody authenticates mutually, every transmission, or even every session, just once upon entry…then you can sit on the network *forever*! – and steal 140+MILLION records!): V1.3 allows you to authenticate *just once initially*, and then never again. This is utterly mysterious, because *there is nothing new in the V1.3 authentication techniques*!

But, it certainly is 'Faster', yet *exponentially less secure*. But…the 'More Secure' is touted, so what, exactly, does *that* mean?! It means that V1.3 has removed some of the older, no-longer-supported-because-they're-instantly-compromised encryption algorithms. This, of course, means that there wasn't anything added to make it more secure, it's just more secure because you can't use some insecure components. [This is akin to saying that you made your house More Secure by throwing away all those extra keys you've accumulated over the years that didn't fit the front door. *WHAT?!?!*]

We offer Qwyit™ as the solution to current methods' inability to be universal: world's fastest, smallest, most flexible and provably secure authentication and encryption protocol. For more on how we provide these features and benefits, see Qwyit LLC; go to www.qwyit.com.

Multidimensional Networks

As evidenced by newer trust models (blockchain, etc.) and the growing communication hardware capabilities (Bluetooth, NFC, etc.), end-to-end network participant authentication and message security in multidimensional networks simply don't work using 'traditional security.'

For instance, a participant in a bank transaction may use their phone, connected to their wireless router by Bluetooth, connected to their wired router, connected to their ISP, connected to a backbone, connected to the bank's entire device architecture, finally connected to their account database – all by different applications, communicating at different OSI levels, with incomprehensible settings, most of which aren't accessible by the user, who has a username/password for their authentication, while the actual application message has an increasing number of network connections prior to the destination – each that should require and perform their own authentications.

This, of course, is the exact definition of the 'end-to-end security problem.'

The 'original digital network' where asymmetric keyed authentication algorithms and single-keyed encryption ciphers were developed no longer exists – one device logging into a single machine; now there are routers, servers, databases, clouds! It is apparent that network authentication and encryption are never going to be able to continue to use these complex, inefficient methods *because they don't work from end-to-end.*

We developed QwyitTalk™ to replicate/replace TLS and provide an improved trust model and implementation for software-based secure communications (where 100% of digital security is practiced). In spite of the substantial improvements offered, Qwyit LLC has concluded that the dire and deteriorating current state of online security will never be adequately addressed until the 'multidimensional network security discrepancy' is solved: users are anonymous, except to their destination, while their communications need 100% provably secure digital transport.

The perfect example of this conundrum is Bluetooth: a wonderful device communications capability – disparate devices connecting to each other to communicate. Great stuff! Invented as a new, multidimensional network capability – variable, on-the-fly, both short and long-lived, etc. Except, it turns out, the owners and/or content should be authentic and secure. As Bluetooth has expanded and been

used in new paradigms, it's become obvious that 'bolting on' current security methods is a complete and utter disaster: See this page for all the phases, modes, levels of Bluetooth security – astoundingly inefficient, insecure and, well…useless. (Like the un-heard tree in the forest, if all the users, developers and experts can't tell from looking at a Bluetooth implementation how its 'security is done', *is it secure*?!)

Conclusion

We need new security that can solve both these problems:

Truly universal constant, mutual authentication and provably secure encryption that fits everywhere across all networks, is always on, and works the same way – everywhere!

Since the Qwyit protocol was designed for flexible implementation, along with our other products that compete directly with current protocols – and improve them by an order of magnitude – we're introducing QwyitChip™ for hardware and QwyitSDK™ for software as two universal, identical operating data encryption products.

They perform our QwyitCipher™ (QC™) algorithm for 100% provably secure, uniquely keyed data security. All any digital developer needs to do is perform two simple steps:

1. Install either QwyitChip™ or QwyitSDK™ into your device, product, application
2. Place one single programming instruction: Get a Key, Call QC™ (it even rhymes! :)

And since we understand the ever-evolving multidimensional network, we've developed and are introducing a new trust model and method for the input into that simple instruction: Get a Key.

QwyitChip™ and QwyitSDK™ solve the Universal problem: they fit *everywhere* and *anywhere* on *any* network.

Now there's QwyitKey™ for universal operation: *every implementation can generate their own anonymous keys.*

We have the entire package that solves both the universal and end-to-end security problems: a universal hardware and software method that **Fits Anywhere, Works Everywhere**.

Why QwyitKey™

The Qwyit™ protocol provides authentication (embedded) and data security (stream cipher) for communications traffic based on a secret-key system. Qwyit™ is available now in its base implementation with the one-call QwyitChip™ and QwyitSDK™: wherever they are installed, the controlling software application needs to perform only one key management task: Get A Key – then call the QwyitCipher™ method residing in totality, identically, on those two products, with that key and the message. It will be sent to the application designated recipient 100% provably encrypted (data secured).

The authentication is entirely dependent on the provided key: what is its purpose and what trust model created, managed and delivered it?

The key is either a system level authentication token or simply a per-message encryption key. The key's origins and management – and therefore its authentication value/metric – can be of any type; including, but not limited to:

- Current PKI protocols (e.g., TLS) where the authentication is handled by PKI and the encryption is our QwyitCipher™ instead of AES
- Our own QwyitTalk™ 3rd party key management system which replaces PKI and TLS, providing mutual, constant authentication and encryption
- Self-managed systems, uniquely designed and created using any aspect of our Qwyit™ protocol; each system defining its authentication and data encryption metrics. [Unique designs can certainly use other protocols as well.]
- This new QwyitKey™ Authentication Service, for applications needing a simple, fast, self-managed, independent-trust key distribution service for authentication and encryption

The first two have all the pertinent available information necessary to articulate those system's features and benefits – the third item should provide the same – and QwyitKey™ is being introduced here.

To be Authentic, there's got to be Trust

Both PKI and QwyitTalk™ follow the 'traditional trusted 3rd party' concept: They provide authentication through a registration process, and extend it into the messaging architecture. The registrations include some form of participant identification and the extended trust of the registration is accomplished by mathematics (in PKI this is theoretically sound; in QwyitTalk™, it is provably sound.) The authentication is then used to share an on-the-fly unique encryption key for data security (in PKI, this is a less-than-perfect encryption algorithm; in QT™, it is the only provably secure perfect encryption algorithm.)

While we noted in the Introduction that those PKI systems will never meet the universal security need, their existing implementations can be used to deliver encryption keys to our QCipher™ products. Also, while QT™ is available as a drop-in replacement for those PKI systems, it can provide both authentication and QCipher™ encryption keys. But both of these require an identified registration (a certificate purchase in PKI; in QT™, a Verified Setup) – and that's a great part of the multidimensional network's end-to-end security problem: they don't work like that.

They work like this:

"Hey Alexa, text my mom." I am sending an authentic message, through Alexa, which is connected to my wireless network. Which will send a message to Amazon, to my Mom's Alexa device – or after checking that there isn't one, to her phone service's texting protocol/server…finally to her device. All of those 'places' should be able to authenticate and secure the message as they each individually need, and as the controlling applications dictate.

There are a great many unsolved problems with the 'end-to-end' security on these networks, dependent on the sender and/or the destination:

- Some of the networks want to know who sent the message – because they don't perform their own authentication/security, so they think they need to know!
- Some of the networks want to 'add' to the existing bloated bandwidth of the already authenticated/encrypted message – and they can only do that with current methods, making the traffic so large, it falls over

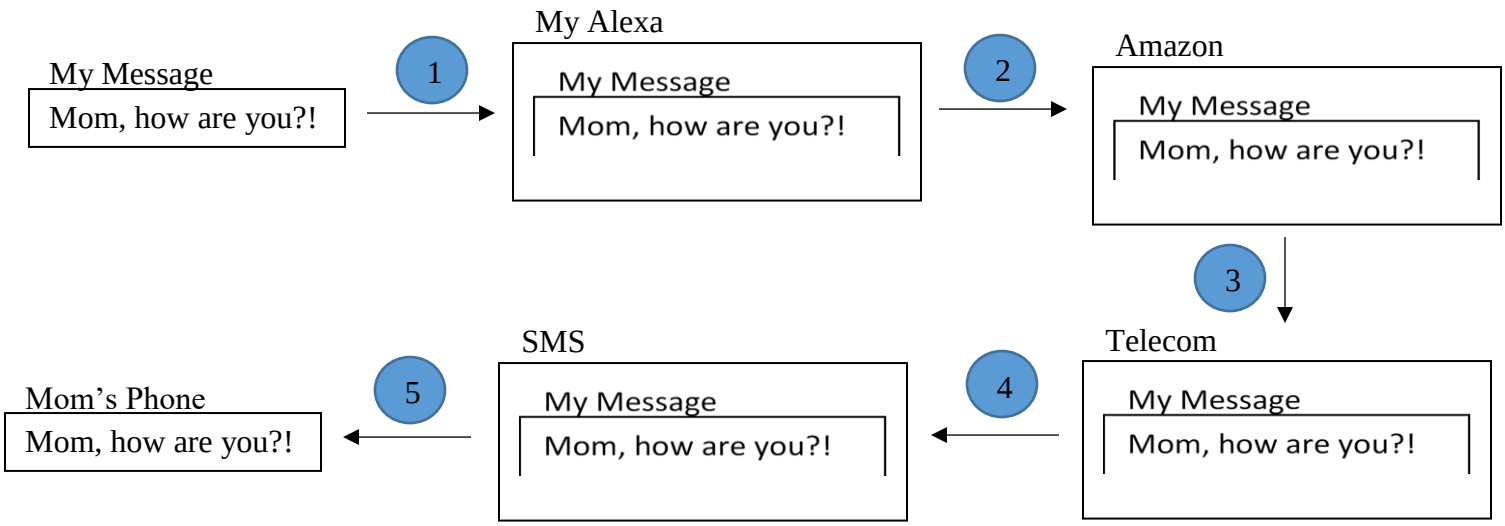
- Some of the networks use 'other' security methods, becoming incompatible with the message
- As these network connections change/update/multiply, so do these problem types

Since it is impossible to 'solve' these individually (which is attempted, but obviously fails miserably whenever there's any future 'change', if the solutions work at all from the start), what's needed is:

A simple, universal method to Get a Key, Call QC™ - *for any traffic, anywhere, such that all individual network's authentications/encryptions are self-managed, and independently trusted everywhere*

It makes no difference *what traffic* is processing, each multidimensional 'place' can self-manage, passing it 'down the line' authenticating, encrypting/decrypting, such that *the message is trusted from start to finish.* For example:

Multidimensional QK™-enabled Network



1. I create a message by texting my Alexa from my phone (I'm lazy…in bed, and Alexa is downstairs!). This message would be QwyitKey™ authentic/encrypted by the Alexa-app on my phone. It's sent to My Alexa – where it is QwyitKey™ bundled in total by the Alexa control app to send it to Amazon (as the owner/servicer of all things Alexa)

2. My Alexa sends it to Amazon, which knows my device, and can decrypt it back to My Message (still QwyitKey™ secured). Amazon's control devices can QwyitKey™ authorize/encrypt the message all over their network, ending on a communications switch, sending to the appropriate Telecom to get to my Mom
    a. As part of the Alexa control app (either the version on my phone or the Alexa box, or both) – they 'know my mom' as one of my contacts – they know the destination number and can manage how to 'get to the right Telecom' 'place' that will deliver the message

3. Prior to, or at, switch delivery, the Amazon QwyitKey™ wrap is decrypted (having assured its network of an authentic, secured message, from start to finish); then it is QwyitKey™ authenticated and secured by the switch to enter that network

4. The devices in the telecom network can QwyitKey™ authorize/encrypt throughout their network, maintaining 100% provable security, wrapping the secure message into their own QwyitKey™ authorized/secured encrypted-SMS message.

5. The final QwyitKey™ authorization/encryption occurs in the sending device out to Mom's phone where the sending device removes the last of their networks QwyitKey™ bundle. Then, on Mom's phone, there is a controlling app (any destination texting application that is QwyitKey™-enabled) that can finally QwyitKey™ authorize and decrypt the text message as coming from Me – using the original QwyitKey™ credentials that only her and I share

OR

My original encrypted message is trusted all the way from end-to-end, simply passed along in its original state. In some networks, this is not only possible, but preferred. The entire purpose of QwyitKey™ authentication is its flexibility in meeting whatever network requirements exist.

Quite obviously, this can't be done using PKI certificates (otherwise it already would be). And while it is more than possible to implement QwyitTalk™ to accomplish this – real-world dynamics make it difficult to manage the identity of the components. The network is routinely changing, both from a traffic and management aspect, as well as physical network devices and components. It is this real-world dynamics that defines 'end-to-end'; and there are some crucially important features of the security that must exist when dynamically applied:

1. It must be anonymous
    a. The devices and management won't be, but the security *has to be*, otherwise it will completely and instantly overwhelm itself: the extra layer of 'identity' for security purposes is not feasible – establishing it to get 'keys', maintaining it to use those keys, and re-configuring it to update/change those keys, etc. On top of network configuration and management of the devices themselves, another identity layer demands too much of the installation, maintenance, programming and user bases. This is evidenced by all the crime, incompetence, mismanagement, user confusion and continued failure of applying security to multidimensional networks
    b. This does NOT mean inauthentic
        i. Network management is performed by many different controlling applications. It is the intention of Qwyit's QCipher™ product line to streamline, standardize and simplify performing authentication and encryption by providing a universal capability that can be put everywhere – and it is our QwyitKey™ authentication service that allows those controlling applications to manage the authentication aspect of device security in one simple programming call: Get a Key, Call QC™. *Those device controlling applications already have and utilize inter-network connectivity methods; that is where QwyitKey™ will be called.* Which is why the *security* identity can be anonymous for network implementation and QwyitKey™ service membership, while it is *not anonymous within* QwyitKey™ key sharing requests and network membership (the participants sharing the same QwyitKey™ keys know each other), and therefore *authentic*
            1. The methodology for how to manage *network device membership would then be the most vulnerable security juncture.* Since *any* device can anonymously join the QwyitKey™ service; it is a simple thing to then insert a somehow-fraudulent device (a repeater, a replacement, etc.) into the

network if the device membership methods and controls (the software, etc.) are insufficient, inadequate – somehow easily compromised.

2. **The QwyitKey™ authentication service will provide 100% provable assurance that the requesting OpenID is the one contacting you. You (or your device) will then have some other means to assure yourself that that OpenID is being used/controlled by the anticipated Sender (or their device). This is the definition of Independent Trust: the authentication of the communication is assured by the QwyitKey™ Service, the communication ownership/authentication is assured within the context of that particular communication, network, etc. (which may be a part, or the whole). This is the job of the controlling application**. In multidimensional networks, hardware management controlling software will perform the job of assuring messaging context/pathways (which devices, switches, apps, etc.); while QwyitKey™ assures each called Authentication Request.
    a. For example, by pre-shared device lists (such as in a single car's IoT control software for all its sensors), by P2P established relationships (the car to the manufacturer), or other financial, interdependent relationships (autonomous driving capability to a car-loaner service.), etc.

c. Independent Trust is the only trust model that allows both existing and new networks the ability to 'bolt-on' or 'design-in' competent, understandable security
    i. Existing networks now have a simple process for implementing meaningful, useful and properly functioning security: Wherever our traffic goes, put a QCipher™ into the architecture (hardware or software), and add the one-command Get a Key, Call QC™ to the controlling apps – then we'll have trusted end-to-end security *while we handle it*, regardless of its state when we get it and finish with it
    ii. New networks, presently completely bewildered and overwhelmed with using/managing/integrating current security methods (so security is either left out, poorly integrated, or unusably fiendishly 'expert-driven'), have the same simple design step as existing network proliferation: Wherever our traffic goes, put a QCipher™ into the architecture (hardware or software), and add the one-command Get a Key, Call QC™ to the controlling apps – then we'll have trusted end-to-end security *while we handle it*, regardless of its state when we get it and finish with it

2. It has to be fast
    a. Performing multiple layers of authentications and encryptions must be done at transmission speed
        i. Qwyit's patented techniques deliver this
3. It has to be easy
    a. Get a Key on-the-fly, use it
        i. QwyitKey™'s one-step authentication request guarantees this
4. It has to be secure
    a. Qwyit's fundamentally unsolvable underdetermined mathematics proves this

How QwyitKey™ works

All QwyitKey™ interaction comes through whatever QwyitKey™-enable Application one is running for the network they are joining. QwyitKey™ has two processes – a one-time anonymous registration, and a minimally-one-time Authentication Request.

1. Anonymous Registration: Anonymously join the QwyitKey™ Authentication Service (community) – Perform a simple, *Anonymous* Set Up (ASU) to obtain a unique OpenID and QwyitKey ASK (QwyitKey™ Authentication Service Keys)
    a. See the Qwyit™ and QwyitTalk™ Reference Guides for more information and different versions of VSUs – the QwyitKey™ guide provides an ASU; but the mission is identical: to 'securely' give the requestor an OpenID, QK, EK (these are the ASK)
        i. 'Securely' is defined by the system where the registrant belongs
            1. If it's enough to trust HTTPS TLS transport, or
            2. ASK needs to be multi-channel distributed (as in the QwyitTalk™ VSU), etc.

    The basic steps:

- Make a request to the QwyitKey site to join (which will initiate the ASU)
    - The ASK is delivered in the 'secure' manner as defined by the registrant system
    - Participant stores ASK credentials
- The participant will send an initial QwyitKey™ Confirmation Message
    - This includes the Sender's Verified ID number
        - This is self-generated, and now held by the Sender and the QwyitKey™ for verification when communicating with others
    - Also includes the Confirmation Salt number
        - This is generated from the ASK and allows QwyitKey™ verification that this message came from the receiver of the ASK
- QwyitKey™ decrypts the CM and publicly displays/stores the VID associated w/the ASK
        - This is the authentication token used in Authentication Requests

2. Authentication Request: Every P2P (or P2MP (multiple participants)) relies on *participant created and managed* keys, where only a key portion is distributed through the QwyitKey™ hub – and is minimally done once for the lifetime of their participant messaging
    a. The token presented by any messaging participant to the QwyitKey™ hub for secure communication with their intended recipient is never publicly available, is 100% provably never determinable by anyone other than the intended receiver participant, is only a portion of those participant's SMK, is only shared between the QwyitKey™ hub and the initiating participant, and the QwyitKey™ hub has no means to associate anonymous members to their messaging applications nor to capture messages emanating from any of the participants (because they are unknown, and the hub doesn't have that capability)

    The basic steps:

    a) The sending participant self-generates a Qwyit™ Session Master Key (SMK in two parts, SQK/SEK), specifically for communication with this intended Receiver
        - Accomplished in one of several ways – capability of RNG/PRNG creation, and/or PDAF generated from stored RNG/PRNG data (for incapable small devices), etc.
    b) The Sender then submits a secure Authentication Request to the QwyitKey™ using their ASK that includes the Receiver OpenID and only ½ of the key, the SEK

c) The QwyitKey™ replies to the sender in their ASK with a generated for only this session Qwyit Initialization Vector (QIV), and a receiver-only readable Authentication Bundle (AB) containing the SEK, the QIV, and the sender's VID – all twice encrypted for the receiver by their QwyitKey™ membership ASK; the sender can't read the AB (the twice-encrypted QIV doesn't leak any information)

3. QATalk: The two QwyitKey™ participants now communicate by themselves, starting with sharing their full SMK in a QAStart message from the sender who just completed the QwyitKey™ AR

   The basic steps:

   a) The sending participant communicates provably securely with their intended Receiver participant with the first message (the QAStart) including the AB
      i. The remainder of the SMK is calculable *only* by the recipient as outlined in the QwyitKey™ *Reference Guide*
      ii. When received and confirmed by the Receiver, this creates their lifetime SMK (their acceptance is defined by *the content of the message*)
      iii. The SMK may be immediately, or at any time, updated either through a Qwyit™ Nil-communication PDAF, or through another QwyitKey™ Authentication Request

Security Notes

Some QwyitKey™ security aspects:

1. The ASU relies on an initial distribution/join security deemed suitable by the participants (through the requirements of the unique system they belong to)
   a. There are several options for initial distribution including our own VSU methods. The process should be risk/reward analyzed
   b. Where any particular participant has joined under one system and desires to message with another participant from another system (it's possible they had different ASU join processes), it's up to the system's to authorize these types of connections

2. After ASK receipt, all QwyitKey™ participant messaging uses QwyitCipher™ provable security
   a. Knowledge of QwyitKey™ keys cannot be gained from the system in action

3. The issue of trusting the hub not to interfere with individual participant messaging by surreptitiously monitoring, substituting, or performing any type of attack:
   a. Is the same as worrying that the government is watching where every piece of money is being transacted (there is no conspiracy!)
   b. Requires omniscient powers since the hub would have to monitor everyone's individual applications to capture their initial message to their recipient, since the token is not the messaging key, but only a portion of it is included in the Authentication Bundle, created by the QwyitKey™ and only readable by the intended receiver (there is no conspiracy!)

4. The SEK and Receiver's identity are never openly sent

5. The SQK is never sent

6. Verification of the Sender (the purpose of the QwyitKey™ Authentication Service) occurs on every self-generated initial message from one participant to their intended recipient; and all messaging uses QCipher™ authentication and encryption in every transmission
    a. Every QwyitKey™ participant, while communicating with others, on the first message (QSStart) to their intended Receiver, they include their VID, and then QwyitKey™ includes it in the part encrypted by the Receiver's key, so the receiver can compare the two – as well as check the public QwyitKey™ listing. There are no MITM attacks, as the first message to the Receiver allows them to compare the VIDs; and they are assured that the Sender used the QwyitKey™ Service. The *content of the message identifies them to each other*; if one isn't sure of any Sender, disregard this connection and create a new one going back to the Sender through the QwyitKey™

        [Note: As to creating an entire false QwyitKey™, then sending phishing messages attempting to dupe users into performing ASUs with senders, then during any ARs the intended recipients are 'known', then sending false QAStart messages attempting to trick those recipients into communication streams (and somehow stopping the real sender messages, especially the real QAStart), this is a lot of work and is simply defeated by systems constantly advising their participants to check for the sender's real VID at the proper public IP of their real QwyitKey™ (there certainly can be multiple, authorized public versions.) This is the type of charade one sees on thriller movies where there is a con being performed on *one* user; but it's a lot more work than just doing the following:]

        i. This doesn't stop someone if they have completely compromised someone's device by stealing it or just their ASK and VID. This is why the ASK and VID should be PIN-protected in storage – but it also doesn't mean much: QwyitKey™ assures that any received message went through the service 100% securely; identifying/authorizing the Sender is *inside the message – this aspect must be accomplished as well.* But, it's possible stolen devices may have information allowing/helping the fraudulent Sender to fool the Receiver.

8. QwyitKey™ keys should be securely stored (either as Qwyit™ noted, or using 'best practice' techniques)

9. QwyitKey™ is nothing more than an anonymous message authenticator: one can be sure that any message sent or received is 100% provably from that OpenID's QwyitKey™ ASK

10. As QwyitKey™ is an anonymous system, with provable Qwyit™ secure messaging throughout, the obvious attacks are either key-based, or identity-based

    a. Stolen individual keys
        i. While this seems dire, the entire purpose of the QwyitKey™ system is to assure content authentication; *key ownership is contextual authentication and is the purview of the controlling application/network*
            1. Solved by getting another QwyitKey™ ASK; alerted by recipient/network alerts upon key use (to be setup/offered by the controlling apps), or upon 'wrong' ownership contextual use ('Hey, that's not Your Voice! Who are you?!')
    b. All participant's keys stolen (QwyitKey™ attack)

i. While this seems incredibly unlikely (due to QwyitKey™ providing publicly verifiable network key management (click “How Your ASK is Stored!” on the website, for example) – and seriously dire – there are two Qwyit™-only methods for protection to thwart such an attack
    1. All system storage is QwyitKey™ Master Key offset, dependent on current operations staff (This means any keys ‘stolen’ won’t be the correct ones; as well as QwyitKey™ executive management being able to exactly pinpoint any unauthorized ‘insider’ problems)
    2. All QwyitKey™ network operations that allow connectivity will require QwyitTalk™ connectivity – no unauthorized, unknown connections will ever be allowed
        a. This isn’t to say there won’t be problem possibilities from wayward staff operations, but these would be traceable
    3. Should this occur, QwyitKey™ would simply publicly ask all participants to anonymously re-Register
        a. This attack does NOT impact individual QCipher™ keys that have already been established for ALL communicating participants – these remain secure

c. All messages captured for a particular individual
    i. This is ‘The NSA Attack’ – should you be under surveillance this is quite possible (See above false-QwyitKey system note)
        1. Does this assume the interloper has the individual’s keys, as in a.) above?
            a. If so, and they are just listening, they will be successful in eavesdropping
            b. If so, and they attempt to be you, they’ll need your contextual authentication within the networks they ‘are you’
            c. If not, they will be 100% ineffective – this attack is meaningless

d. All messages captured for all participants
    i. This is ‘The Conspiracy Attack’ – fairly impossible, but…
        1. Does this assume the interloper has the all the keys, as in b.) above?
            a. If so, and they are just listening (and somehow they solve the offset protection and are unobserved), they will be successful in eavesdropping (Big Brother is TRUE! ☺
            b. If so, and they attempt to be anyone, they’ll need that person’s contextual authentication within the networks they ‘are that person’
            c. If not, they will be 100% ineffective – this attack is meaningless

11. QwyitKey™ keys
    a. Sizes can be system-specifically modelled
        i. Qwyit™ primitives are based on *any* key length of even-numbered bits, with minimal extension of 1 byte (as time passes and computing power increases, QwyitKey™ keys can simply expand to meet the computational security requirements)
    b. Control hierarchies can be included
        i. QwyitKey™ keys can actually include Access Control within them, not effecting computation
    c. Allow for simple, secure storage techniques (PIN-related offsets, etc. for man-to-machine connectivity w/o need for difficult passphrase implementations – see above, and other Qwyit™ documentation for details)

d. Limit bandwidth requirements (storage, transmission, etc.) to well-defined, easily implemented definitions within any communication specification requirements standards