Qwyit™ Variable Keys

Qwyit™ is an authentication and data encryption algorithm/protocol; see the current version of the *Qwyit Protocol Reference* document, available from Qwyit LLC; go to www.qwyit.com.

Introduction

The Qwyit™ authentication and encryption algorithm/protocol has several unique capabilities not found in any other techniques. One of these is the ability to vary the Key sizes used in the protocol at various processing/application frequencies (such as during a Verified Setup (VSU) or an Authentication Handshake), as well as during use in the QwyitCipher™. This document lists some, but not all, of the possibilities and how the variations are communicated between participants.

The three main areas of flexible key use include Initial, At-Update, and In-Use. The most important aspect of Qwyit™ to keep in mind is the overriding operating principle that Qwyit™ keys can be *any length divisible by 8-bits (a single byte).* This means that with respect to current computational security (brute forcing the determination of any upper-level MEK or MQK), Qwyit™ keys can be any *n-byte* length [16-bits, 8-bytes (64-bits), etc. The current (11/2019) recommendation is for 256-bits.]

Initial Qwyit™ Key Lengths

The first aspect of varying Qwyit™ key lengths is during initial establishment within any single or combination communication network/storage. This is accomplished in a number of ways:

- For a single, static operating environment (consistent device architecture and communication/storage protocols), it is possible to configure different key lengths by user (access controls), by devices (mobile, desktop, server, routers, etc.), by data importance (financial, personal, public, etc.), etc.
    - Operation of the system would be such that the static multi-length key setup would be incorporated into the Qwyit™ processing throughout the architecture; so that all operation *expects* the static/set, but varying, lengths.
- For a multi-operating environment (where there is differing key lengths across ‘static’ included networks – and/or where any network has ‘fluid’ (changeable) key lengths within it over time), it is possible to configure different lengths by the same aspects (access, importance, etc.) using a message and/or establishment flag value for the expected key length
    - This is a public value, as the systems’ various architectures and/or integrations are known and available (for creating new multi-network systems) – there is no secrecy in initial key establishments.
        - The method for incorporating different key lengths at establishment, is to add a public Key Length Flag (KLF) value to the VSC1 message in the VSU. The value would be the length (# of bits), or a known value=length number (e.g., 1=8-bytes, 2=12-bytes, etc. These can be universal (for all Qwyit™ systems), or unique (but published so interacting systems can readily find the appropriate key lengths).

There may well be other, publicly created methods for establishing the Qwyit™ initial key lengths across different networks; and as long as these are published and known, unique, varying Qwyit™ keys can be used in any system.

At-Update Qwyit™ Key Lengths

The unique functionality of 'Endless Keys' in Qwyit™ is performed by using the fundamental cryptographic PDAF primitive, and additionally if desired, the OWC. Both these functions are one-way gates that perform our newly defined *Random Rearrangement* property (whereby two existing random numbers, when pointed into each other, and MOD added together as per the PDAF process, produce newly undeterminable results (without knowledge of the starting random values.)) This process can be performed endlessly without any loss of entropy, or randomness – and is accomplished with NIL communication (no messages are sent). Another version of this 'endless creation' is used in the QwyitCipher™ to produce new, undeterminable, random message keys 'endlessly' to provide unique key bits per plaintext bits. Performing the 'endless' process on a regular basis also allows update of the upper level MEK/MQK – where regular is any of three methods:

- Static – known, repetitive basis (e.g., once every 10 uses, or once a month, etc.)
- Pseudo-Random – at intervals unique to current values in key positions (e.g., at an interval after the number that exists in the 1st and 3rd digits of the MEK uses has occurred (say 56, or E7 with hex transposed to decimal); or any other shared value interpreted as an interval (days, uses, etc.)
- Random – at intervals determined by some system level parameters, directors, content, etc. (e.g., an email is sent by a bank for everyone to change their keys by Friday, by employee ID number, etc.)

It's quite straightforward to see that at *any of these At-Update* 'endless' key updates, the key lengths can be changed as per any of the same three types, this time for determining key length:

- Static (e.g., a shared table indicating length changes longer and/or shorter, a known pattern that is cycled through, etc.)

  Note: At-Update and Initial Qwyit key lengths *can be lengthened or shortened*; but the following In-Use key lengths are recommended *only to be shortened*. This is because At-Update and Initial are either simply created or PDAF created, both of which can lengthen a key; In-Use key length changes are performed within the QwyitCipher™, and in order not to introduce any performance degradation, a simple trim is performed based on the existing standard (full) length. A PDAF instruction set *can be introduced* within the QC™ if desired allowing key lengthening as long as the (albeit extremely fast and minor) performance hit isn't an issue.

- Pseudo-Random (e.g., to lengths defined in changeable shared values (digit position values MOD 100 to keep them within a range, etc.))
- Random (e.g., to lengths determined in some system level parameters, directors, content, etc.)

The main consideration of these changes is maintaining a sufficient key length limiting brute force attacks as per the security desired (strength based on content) at the time of use (strength based on current computational capabilities.)

In-Use Qwyit™ Key Lengths

In relation to all of the above, the same key length variability can be accomplished *within the QwyitCipher™ encryption/decryption process*. This can happen in two ways (including both of these at the same time):

- *At the beginning* of any discrete message (defined by any system as to what that means: a session between two participants, or each individual transmission thereby using the same OR to create an endless series of child Message Keys, W; or any other system-defined begin-end)
    - This is performed by including a system-added KLF at the beginning of the discrete message (where each discrete message will use the differing key length indicated by the KLF).
        - For QwyitTalk™ keyed systems, the KLF is simply an added value to the QTQS message [The system-defined use of variable-length in-use keys would be an additional/indicated meaning in the VSU KLF]
            - How this value is determined is at the discretion of the system owners/management
        - For Q-ANON™ keyed systems (and any such 'use this key' based systems), when the Authentication Request (and accompanying Authentication Bundle) would include a KLF that may be either a system-level key varying definition, or participant-only key varying definition.
            - These additions are particular to any system choosing to implement QC™ variable in-use, at-beginning, discrete message key-lengths – and the definitions would be at the discretion of the system owners/management
- *During* any discrete message
    - This is performed by adding a KLF variable to any QwyitCipher™ implementation. There are two distinct ways in which the *During Use* case may update/change the to-be-used Message Key length
        - Method A (In which each single called message data end/dec is operated on using a static key length, after a 1st-pass 'standard' key length): In both the QwyitChip™ and QwyitSDK™ universal implementations, this would include adding the KLF as an input variable to the QC calls
            - The value would be the bit-length determinant of *the length of the NextR value in Step 5*. The first pass would be the 'standard system-defined length' (e.g., 256-bits), and the follow-on messages would use a KLF-defined length by 'chopping' the MOD result of NextR (one cannot add to the key lengths without the introduction of a PDAF instruction; the system-defined length is the maximum). Once this new key-length is defined and determined, all following passes through the QC™ will be at this new length (as the length of NextR determines the Message Key's (W's) length.)
        - Method B (In which each unique pass of each created next-n-bit enc/dec is uniquely key-length determined): If both QC™ and QSDK™ universal implementations, this would include a trim of Step 3's creation of the W. The new trimmed length (W') would be used in the Step 4 encryption/decryption; but the full W would be used in Step 5's NextR creation (such that the next W's creation is always at the system-defined length, but possibly trimmed for it\s next enc/dec use.)

- The amount of trim can be determined by any system-defined variable value selection (e.g., using the last byte value in the alphabet A as the number of bytes or bits to trim). This value will change as each NextR pass is calculated.

Note: As noted previously, these In-Use key length changes are simple trims of internal QC™ calculated values – being careful to retain the standard-length values for next-round calculations, otherwise the keys may 'disintegrate' down to easily brute-forced values. It should also be noted that there are other trims available within the cipher; as well as adding additional PDAF and PDAF/OWC instructions for key lengthening – at various stages of cipher calculation, using different inputs (A, W, NextR, QK, EK, etc.)

Security Considerations of Qwyit™ Key Lengthening

There are two major considerations for any impact on the security of Qwyit™ when varying the key lengths from Standard Recommended Length:

- For Initial and At-Update, shortening the length of any easily known key value such that manageable brute force attacks will reveal/leak information *must not be done*
- For In-Use key variation, since QwyitCipher™ produces OTP-secure (provably unbreakable) output using the standard length, the only reason to perform key variation would be system-limitations/variability; otherwise, In-Use variation is meaningless

Lastly, just because Qwyit™ *can vary its key lengths* in multiple ways, the only reason to do so is content-based separation, creating 'security levels' (such as 'Secret', 'Top-Secret', 'Eyes-Only', etc.) Where this is applicable is debatable, in light of Qwyit's™ provable security at any key length. This document simply presents another aspect of Qwyit's™ superior design.