QwyitTalk™ Overview

The QwyitTalk™ Authentication and Encryption *Service* (QwyitTalk™ *Security as a Service*) is a direct, next generation replication and enhancement of the current, only global secure communications framework: Transport Layer Security (TLS). Qwyit provides the same features, benefits, authentication and data security (stream cipher) for communications traffic using the Qwyit Directory Service (QDS) key store. QwyitTalk™ (QT™) 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. *The QwyitTalk™ Reference Guide* and client demo APIs are available as well; go to www.qwyit.com.

Why QwyitTalk™

In order for cryptography to look forward to a better future, it's helpful to learn from the past. The single most important development since 1917, when the 'only' perfectly secure encryption was created (Gilbert Vernam's One Time Pad cipher), was the 1970's Public Key systems. From *The Code Book*, by Simon Singh, Doubleday, 1999; pp. 279-80:

The story starts in the late 1960s, when the British military began to worry about the problem of key distribution. Looking ahead to the 1970s, senior military officials imagined a scenario in which miniaturisation of radios and a reduction in cost meant that every soldier could be in continual radio contact with his officer. The advantages of widespread communication would be enormous, but communications would have to be encrypted, and the problem of distributing keys would be insurmountable. This was an era when the only form of cryptography was symmetric, so an individual key would have to be securely transported to every member of the communications network. Any expansion in communications would eventually be choked by the burden of key distribution. At the beginning of 1969, the military asked James Ellis, one of Britain's foremost government cryptographers, to look into ways of coping with the key-distribution problem.

Envisioned 'insurmountable key distribution problems with secret key systems' led to the development of these Public Key Infrastructures (PKIs), and the only current global secure communications protocol based on them, Internet communication using TLS. TLS is a 20-year effort based on the fundamentally sound, but 'environmentally challenged' PKI methodology. 'Environmentally challenged': "*The extra latency and computational costs of the full TLS handshake impose a serious performance penalty on all applications that require secure communication.*" – *TLS, Networking 101, Chapter 4*

There are several other issues with TLS, most of which are 'solved' with a patchwork of upgrades, extensions, expert control, additional hardware, etc., but all unfortunately introduce more cost, complexity, inflexibility, incomprehensibility, increased bandwidth, and even decreased security when attempting to 'solve' these problems. In short, TLS has reached the full extent of its capabilities, leaving its one global secure communication application lacking, and never even reaching any others: Internet of Things, autonomous vehicle control, multi-cast entertainment, etc.

The problem still remains: *Global, cross-application, multi-platform, simple key distribution and management for use in provably secure, private, authentic communication.*

Since there has been only *one* successful global security model (TLS), why not replicate, extend and enhance it in order to meet and deliver the system properties that will solve Mr. Ellis' still-remaining problem of widespread secure communication: Performance (Speed). Simplicity (user and implementation). Flexibility (trust, platforms, application). Efficiency (size and bandwidth). Security (privacy, authentication, provable basis).

Let's build a system, that works identically to TLS's proven business and technology model, that:

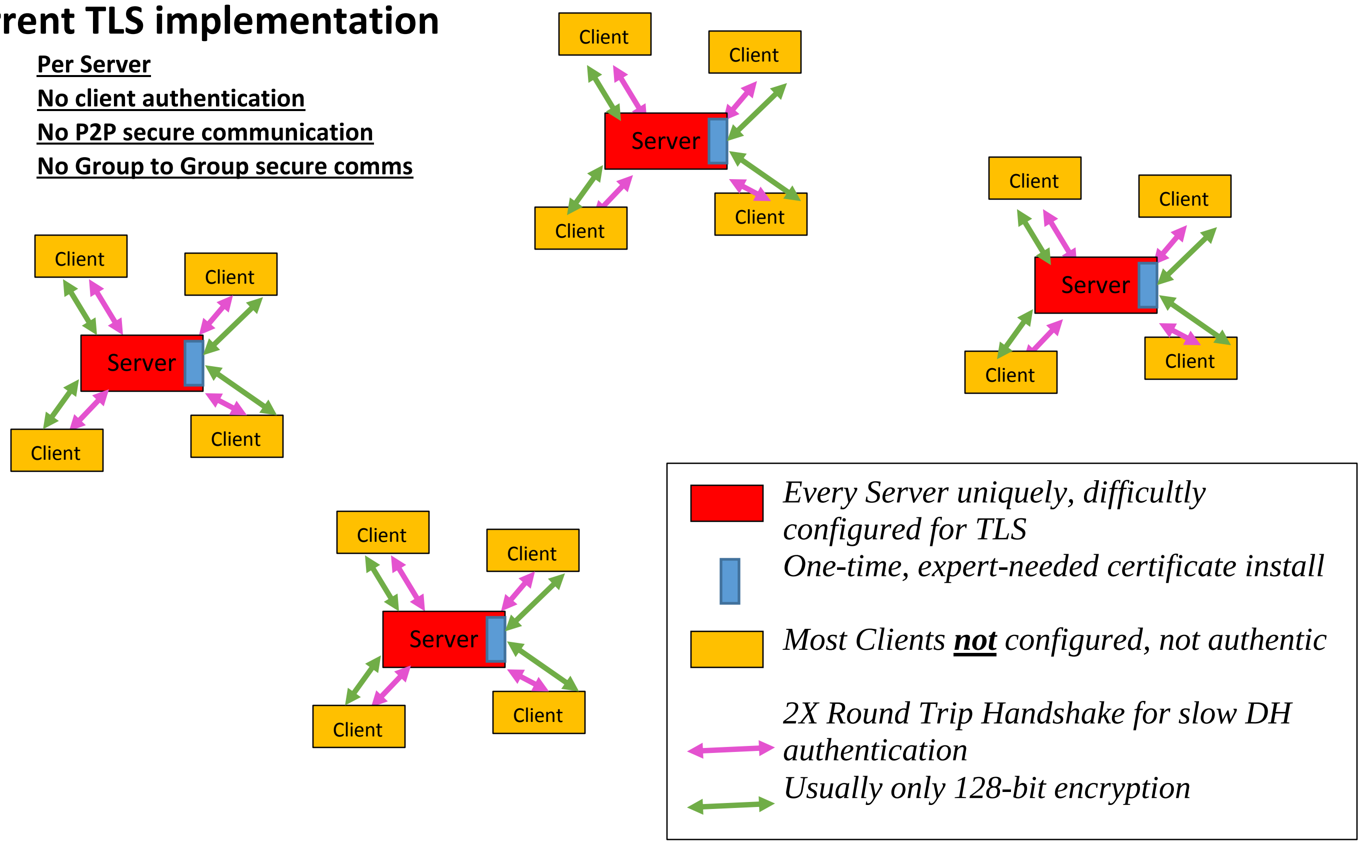
- Performs within the network tolerances without degradation, without new processing power
- Is easily placed into existing code bases, hardware chips, communication protocols
- Doesn't require the end-user to 'do stuff they don't understand'
- Expands, contracts, segregates and includes different trust models, across multiple platforms and protocols, can be integrated the same way in different applications
- Is small enough in code, firmware and hardware not to impede or limit anything already accomplished
- Uses the same provable basis of security as last century's leap forward: mathematics, not computation, or theoretic
- Is identical in every way to TLS (Perfect Forward Secrecy (PFS), authentication, encryption, integrity and replay prevention)

QwyitTalk™ is next generation TLS, and the answer to globally secure widespread communications because of the unique underlying methods.

How it works

Current TLS implementation

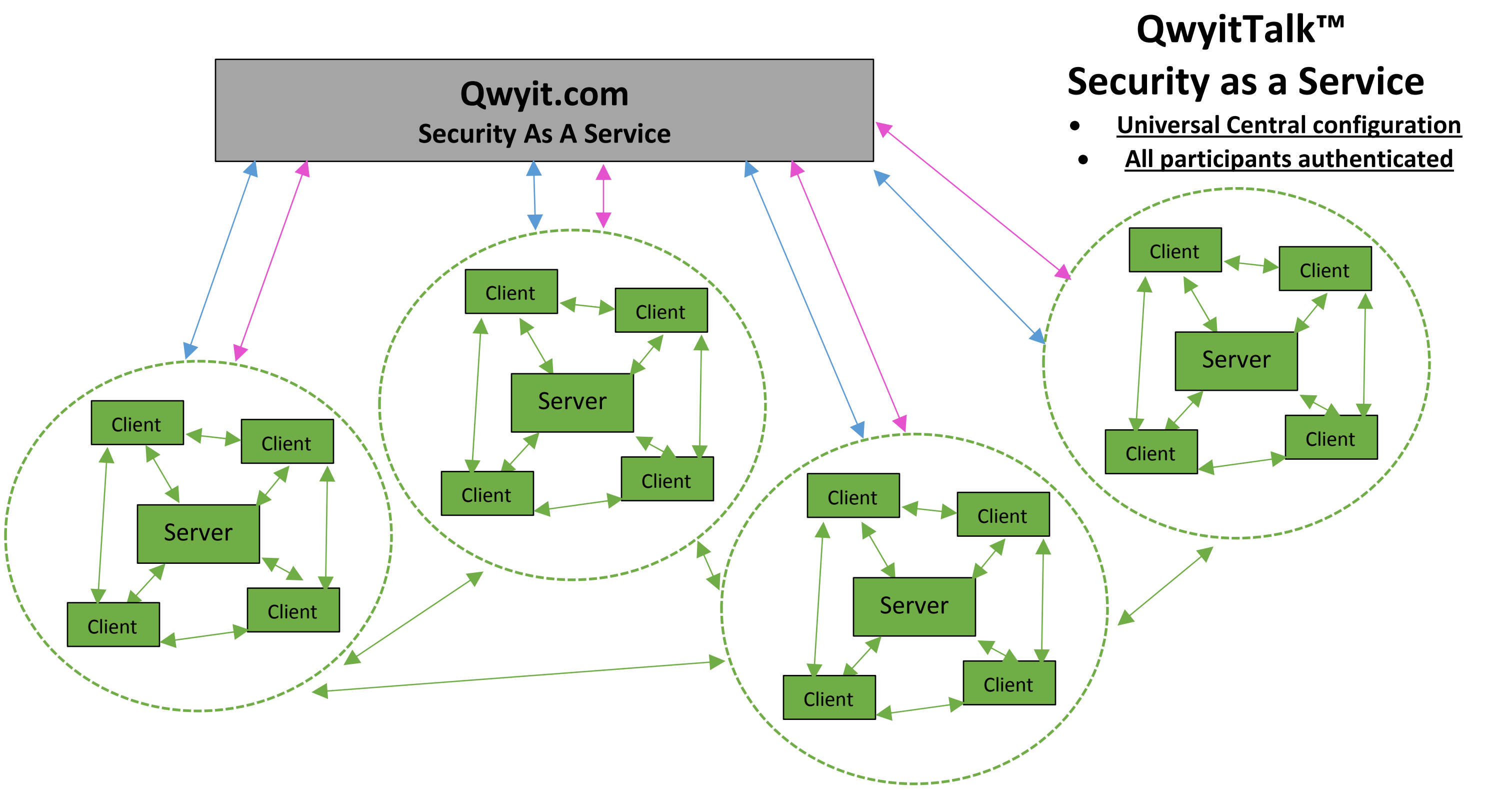
- **<u>Per Server</u>**
- **<u>No client authentication</u>**
- **<u>No P2P secure communication</u>**
- **<u>No Group to Group secure comms</u>**



🟥 *Every Server uniquely, difficultly configured for TLS*
🟦 *One-time, expert-needed certificate install*

🟨 *Most Clients **<u>not</u>** configured, not authentic*

↔ *2X Round Trip Handshake for slow DH authentication*
↔ *Usually only 128-bit encryption*



⬜	*Qwyit provides uniform, managed TLS Service*	↔	*All do uniform, simple one-time Verified Setup (VSU)*
🟩	*All participant communications uniformly enabled and authentic*	↔	*All do Authenticated Handshake (AH) per session (1X RT, fast Qwyit exchange)*
		↔	*All QwyitTalk securely in 0-pass authentic encryption (256-bit, world's fastest cipher)*

Why Not Build QwyitTalk™ With Existing Techniques?

Based on the pictures of the current TLS vs QwyitTalk™ systems, it's obvious that the benefits of a unified, centralized approach include increased/improved simplicity, flexibility, security, efficiency and performance. But it's also an obvious inquiry: why can't one just re-arrange the current TLS into the centralized QT™? The short answer is: It won't work. The quick evidence: it would already have been done! Here is a short list of more technical reasons:

- In TLS, there is no connection, no pass-through evidence, no mathematical relationship between Authentication tokens (keys, certificates, etc.) and Encryption keys
    - This results in a minimum of two passes to convey/connect an Authentic user (presented with, and verified by, an Authentication technique) as the same Secure user (encrypted traffic)
        - This dual-nature, disconnection means all the P2P exchanges require either inefficient multiple transmissions, or convoluted 'already-in-place' techniques (leading to insecurity) – See Whatsapp, Whisper, etc. for their 'cheat' attempts to rectify this underlying disconnection
    - There must be two different token storages, management systems

- The techniques require too much ‘system energy’ (computing power, time, number of messages, message lengths and content, etc.), losing the centralized efficiency

- Just using a single token, such as managing AES cipher keys w/o an accompanying authentication technique and tokens, and attempting to perform the same QT™ P2P cipher exchanges won’t work:
    - There are no authentication properties associated w/cipher-only keys
    - There are no PFS techniques to update the keys
    - Block-ciphers are order of magnitude slower (‘system energy’ issues remain)
    - The AE AES modes of next-block ciphering aren’t related mathematically – only bit-fiddled – providing no true authentic forward block encryptions

In summary, *current techniques are at their end-of-life: it is time for a new, better approach!*

Centralized Trust

Qwyit™ LLC can hear it now: *A single, worldwide entity managing all these keys?! – Talk about Big Brother!!! This is an absolute NONSTARTER!!!*

Here’s the thing: Do you trust Amazon? Google? Facebook? Twitter? Of course not – not any more than you’d trust QwyitTalk™. But you give them ***real, personal data***. What data are you sharing w/ QT™? ***NOTHING***. There is no Centralized Trust issue with QT™. QT™ doesn’t want, need, hold or in any way require end-user trust:

Centralized management of System End-user Authentication does not entail any compromise of Individual End-User Authentication as defined by the Owning Network’s entity and authority

Meaning: When you talk on the phone to your intended party, only you and your intended party trust each other. You don’t have to *trust* QT™, you just have to *use* them to make a secure call. What’s the difference? It’s simple:

You don’t trust the other fans at the game not to steal your money as you pass it to the vendor at the end of the aisle for your drink – but you have to use them. The vendor checks your money to see if it’s real, and you get to decide if the drink is real; and certainly, none of the fans handling your money could ever use it as if was theirs! *None of the people handling (managing) the transaction need to be trusted – you won’t get a drink if the vendor doesn’t get the money, and there’s no method that the handlers could use to take either!* This principle of ***pass-through trust***, this primary property, shows that System Authentication (The Fans) don’t compromise the Individual Authentication (you!) as defined by the Owner Network (Vendor). You simply have to use them. And what’s the more efficient network? Right – going to the vendor’s large drink shop in the corridor: *that’s* QT™.

The bottom line is that multiple, interconnected QT™ centralized key management systems can, and will be built. All they do is *transact trust*, they don’t hold it, require it, ask for anything from any participating client. They are truly just a partner in your client-to-client, P2P authentic and secure communications, and they *can not* interfere in your connection – something people *attribute to* Google, Facebook, et al, but in reality they don’t provide trust. QT™ does; and the benefits the central service provides are truly revolutionary in finally delivering a global, digital security system.

NOTE: In TLS, you trust centralized authorities; you just may not know exactly how it really works. You don't actually *trust* Amazon, you *trust* that they properly acquired a Certificate from a Certificate Authority – and are managing and using it properly. Right: *those CA's are centralized trust*! Comodo, GoDaddy, Symantec, etc..*you trust these people?* Goodness: these folks *have control over your* **data.** QT™ only brokers a pass-through transaction with absolutely nothing of value in it – and then certifiably, instantly deletes their participation. ***This will be publicly transparent, and you'll be able to verify 3rd party certification of this deletion***. Your trust in us is nothing more than that in the fan next to you passing your money to the vendor.

Build QwyitTalk™

Suggested QwyitTalk™ Prototype Development Plan

1. Identify an appropriate client/server area for demonstration, including the following properties
    a. Best fit development platform
    b. Best fit open source code, with easy integration of QT™ code and libraries
    c. Best fit benchmarking against *normal* operation (not any 'secure' versions, as these may/will have special implementation guidelines: compare QT™ to insecure)
    d. Best fit optimization so as to accomplish claimed benchmarks (QT™ should add only a small overhead without any hardware or software modifications to insecure operation)
2. Update Server, with the following capabilities
    a. Listen on port 4180 (HTTPX)
        i. Perform VSU - Web (HTTP and HTTPS), email and SMS capability
    b. Perform AH
    c. Key storage/update (DB, file, etc.)
    d. A best-of-breed random number generator (NIST approved recommended)
3. Update Client, with the following capabilities
    a. Listen on port 4180 (HTTPX)
        i. Perform VSU - Web (HTTP and HTTPS), email and SMS capability
    b. Perform AH
    c. Key storage/update (DB, file, etc.)
    d. Local version (OS embedded) best-of-breed random number generator (NIST nice)
4. Build installs
    a. Client includes building/creating an SDK that others could use as a guide for insertion into other products, as well as building their own client versions of the same demo product (to see for themselves how easy it is to use/build, as well as independent benchmarks)
    b. The Server build will stay 'proprietary' (even though well documented), as it is envisioned that the server is licensed for others to operate their own independent DSs, as well as federate the trust of our DS to anyone who wants to connect
        i. This means that a follow-on activity (phase) to a successful demo is to define (reference guide like this one), then build an SDK that connects DSs together (how they message for client searches, do a more-robust version of the VSU (more factors of authentication as mentioned in this VSU guide), etc.
5. Test
6. Benchmark

There are libraries available for the fundamental QT™ functions

For the complete technical QT™ implementation details, see *The QwyitTalk™ Reference Guide*