Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: 19 September 2025

M. Montagna
M. von Willich
M.D.F. Aelmans
Quantum Bridge Technologies Inc.
G. Grammel
Juniper Networks

18 March 2025

Distributed Symmetric Key Establishment (DSKE)
draft-mwag-dske-02

Abstract

This document defines a robust, scalable, and future-proofed security protocol for Symmetric Key Distribution without reliance on asymmetric encryption. This document delineates the protocol's specifications, security model, and architectural integration of the Distributed Symmetric Key Establishment (DSKE) protocol.

Note

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Status of This Memo

Copyright Notice

Table of Contents

1. Abstract

This document defines a robust, scalable, and future-proofed security protocol for Symmetric Key Distribution without reliance on asymmetric encryption. This document delineates the protocol's specifications, security model, and architectural integration of the Distributed Symmetric Key Establishment (DSKE) protocol.

2. Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress".

This Internet-Draft will expire on 22 April 2024.

3. Copyright Notice

4. Introduction and background

Symmetric and asymmetric cryptographic schemes employed for digital communication often assume that a hypothetical adversary is computationally constrained. A classic example is the widely used RSA asymmetric scheme, which assumes that factoring a large integer into its two prime factors is computationally infeasible. For an adversary in possession of the factors, the communication between the parties involved would be completely insecure. In particular, asymmetric cryptographic schemes cannot achieve so-called information-theoretical security, and their security will always be mined by computational advancements.

Peter Shor discovered an efficient quantum algorithm for factoring large integers [2], named Shor’s algorithm. When run on a quantum computer, it can break the RSA scheme, Diffie寰滴ellman key exchange, and elliptic curve cryptography, and consequently poses a serious

threat to Public Key Infrastructure (PKI). Although a cryptographically relevant commercial quantum computer is not yet available, a malicious eavesdropper can readily store data being transmitted today for when new breaches of a protocol are developed, or technological advances make existing theoretical exploits practical.

The advancement of classical and quantum computers, together with the development of new code-breaking algorithms, will have an important impact on long-term security, and threaten political, social and economic stability. For these reasons, several governments and international organizations are planning a transition to quantum-safe cryptography solutions, and NIST is currently standardizing post-quantum asymmetric cryptography [6], also known as Post-Quantum Cryptography (PQC).

Besides security, asymmetric cryptographic schemes also come with non-negligible operational and computational costs. For example, in SD-WAN networks, before a pair of routers can exchange data traffic, they establish an IPsec connection between them, which they use as a secure communications channel. In a traditional IPsec environment, key establishment is handled by the Internet Key Exchange (IKE) protocol [5]. IKE first sets up secure communications channels between devices and then establishes security associations (SAs) between each pair of devices that want to exchange data.

While a lot of effort has been invested in the standardization of asymmetric cryptographic schemes and their integration into the Internet, symmetric key distribution systems did not achieve a similar level of international standardization. The Distributed Symmetric Key Establishment (DSKE) protocol is introduced here to allow for interoperability in symmetric key distribution systems. DSKE relies on pre-shared random numbers between DSKE clients and a set of so-called ’Security Hubs’.

Any group of DSKE clients can use the DSKE protocol to distill a secret key from pre-shared numbers. The clients are protected from compromise of Security Hubs by a secret sharing scheme that allows the creation of the final key without the need to trust individual Security Hubs. Precisely, if the number of compromised Security Hubs does not exceed a certain threshold, confidentiality is guaranteed to DSKE clients and, at the same time, robustness against denial-of-service (DoS) attacks is assured.

DSKE can be proved to achieve information-theoretical security, and a very high level of scalability, both discussed in detail in this document. Other advantages include computational efficiency and potentially the limited amount of code needed to implement DSKE, reducing the surface attack.

Furthermore, the intrinsic trust distribution of the DSKE backend provides a further level of redundancy against network failures and removes the single point of trust. The DSKE system can be used in situations where asymmetric cryptographic schemes are not suitable, for example:

Quantum-secure communication, including government communication, national security systems and critical infrastructure, which may have security requirements that asymmetric cryptographic schemes cannot deliver. National security agencies in many countries recommend the use of symmetric Pre-Shared Keys (PSKs) instead of or in addition to asymmetric public/private key pairs to provide quantum-resistant cryptography.

Large meshed networks, where the use of IKE and PKI can become a computational bottleneck, or can require the management of a large number of certificates, which increase network complexity and reduce scalability. Such networks include large meshed SD-WAN and Border Gateway Protocol (BGP) routers, among others.

Cross domain communication, where a single authority or service provider (like a CA) cannot be fully trusted. Instead, multiple parties can establish their own Security Hubs and build an infrastructure in which the trust is distributed among them.

Constrained devices, where key establishment via asymmetric cryptographic schemes is challenging due to the high computational requirements. Examples of these networks include sensor networks [7].

Cloud authentication services, where a customer or cloud service provider wishes to ensure that the connections to specific cloud services are assured to be quantum-secure.

5. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

6. DSKE outline

The parties in the DSKE protocol are the two endpoints, Alice and Bob, who wish to establish an end-to-end symmetric key, and security hubs, which facilitate the establishment of the keys without ever knowing these keys.

The protocol is built from the following elements:

Sharing of PSRD by each Hub

Each Security Hub securely distributes an adequate amount of high-entropy pre-shared random data (PSRD) to each endpoint, with due diligence for ensuring trusted identities, confidentiality, integrity, authenticity through physical security, as demanded by the context and security requirements of the entities.

An AEAD encryption as a potential additional encrypt-and-authenticate security layer for the PSRD delivery, using a separately shared pre-shared key (PSK) between the security hub and the endpoint.

An online activation protocol for acknowledging and mutual activation of PSRD once the endpoint is satisfied that delivery was without tampering or security breach, consuming some of the received PSRD in a cryptographic proof of receipt and integrity.

Key establishment

A threshold secret-sharing scheme 窶a Shamir secret sharing scheme over the field GF(2^8), applied byte-wise for the length of the secret, producing one share per Security Hub. The scheme parameters are: the number n of shares and the threshold number k of shares needed to reconstruct the secret.

An authentication algorithm, which allows Bob to verify that the reconstructed secret is as sent by Alice a polynomial over the field GF(2^128). This authentication is unusual, in the sense that it requires k of received shares to be confidential, rather than some previously shared key. Its cost is that 384 bits (48 bytes) of the shared secret are discarded.

An encryption algorithm for secrecy of each share during transfer from Alice to a Security Hub, and again from each Security Hub to Bob exclusive-or, which is effectively the binary one-time pad algorithm when used with PSRD.

A second authentication algorithm for authenticity and integrity of the message that transmits each share, almost identical to the first authentication algorithm. Its cost is 256 bits (32 bytes).

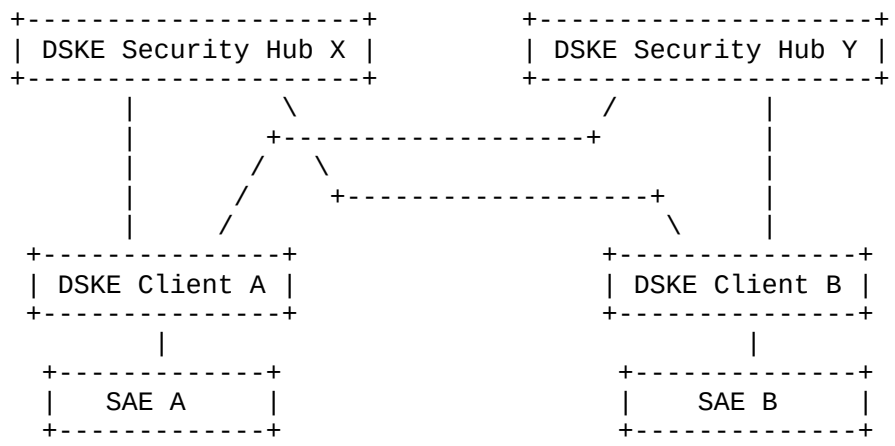
A wrapping AEAD protocol for the messages carrying the encrypted shares, whose primary purpose is anti-DoS, anonymity and confidentiality of control data. The key may be a relatively static PSK. It is not essential to the underlying security of DSKE.

A variation that may be considered is where the PSRD is instead a keystream that is generated at both ends from a periodically refreshed PSK received from each security hub. This trades the theoretical ITS property of the resulting DSKE keys for using only PSKs and limited storage needs, which would be appropriate for small embedded systems, and does not modify the core DSKE protocol or processing.

A secret (the key to be shared, if obtained from a suitable source) can be used to generate shares by Alice using locally sourced high-quality random data, and the above outline applied directly, and this is the most easily understood scenario. However, when no local source of random data is available, PSRD from each of k security hubs can be used directly as its share. The key to be shared and the remaining shares are then deduced, and exclusive-or encrypted as described, the initial k share ciphertexts simply being zeros due to cancellation. Local (potentially weak) randomization of the shares may be added to make all shares look random, but is not needed for security.

7. DSKE System Overview

In a DSKE system there are two primary actors involved: Security Hubs and DSKE clients (or simply clients). A DSKE system usually includes multiple Security Hubs, for reasons that will be clear later, but it can in principle work with only one Security Hub.



Clients join a DSKE system to establish shared cryptographic keys with each other. When a client joins a DSKE system, each Security Hub generates a certain amount of high quality random numbers, referred to as Pre-Shared Random Data (PSRD), and delivers them to the client. Each PSRD set is uniquely shared between the client and the Security Hub that generated that specific PSRD set. Once the client receives PSRD from all the Security Hubs, it is on-boarded in the system and can start to establish mutual cryptographic keys with any other on-boarded DSKE client.

More specifically, when a client wants to establish a key with one or more others, she communicates this to the set of Security Hubs. Each Security Hub, in response, sends a key instruction message to the other client(s), who then use these to distill keys from the PSRD that was pre-shared with them by the Security Hubs when they joined the system. After the distribution of PSRD, which can be done via physical delivery, the subsequent communication between DSKE clients and the Security Hubs can take place over unsecure, public channels.

DSKE clients usually do not consume cryptographic keys themselves, but rather pass them to external applications, referred to as Secure Application Entity (SAE). SAEs can connect to DSKE clients in different ways, which are discussed later in this document. The key established by DSKE clients can then be consumed at will by the SAEs, for example to seed symmetric encryption algorithms or authentication schemes.

In the following we provide an overview of the role of each party in a DSKE system. The next section will then describe the details of the DSKE protocol.

8. Security Hub

DSKE is supported by a set of independently operated Security Hubs. The role of the Security Hubs, in a nutshell, is to: (i) generate high-quality random numbers (PSRD) and distribute those to DSKE clients, while keeping a local private copy; (ii) act on requests from DSKE clients by providing key instruction messages that allow the clients to establish a key.

Each Security Hub is composed by one Security Server, and one or more Local Distributors. The purpose of multiple Local Distributors is to be able to easily provide PSRD to DSKE clients in diverse geographic regions. A Local Distributor generates a large amount of unallocated PSRD as or before it is needed, i.e. PSRD that has not yet been allocated to any DSKE client. The Local Distributor then delivers a copy of this PSRD to its Security Server, while keeping a local copy. The PSRD delivery is usually performed via physical shipping, although other possibilities are discussed later. When a DSKE client requests PSRD from the Security Hub, one of it Local Distributors fulfills the request by allocating a part of the unallocated PSRD to that client, delivering it to the client, and informing its Security Server about which range of the unallocated PSRD was allocated to any given client. The final result is that the Security Server has some PSRD in common with the DSKE client. Note that the Local Distributor does not keep a copy of the allocated PSRD, and PSRD always exists only in exactly two copies.

The Local Distributor, when delivering PSRD to a client, also performs the important task of validating the client’s identity, coupled to the delivery of secret data to be used for later authentication. The authentication of a client to a Security Hub provides, at a later stage, the basis for authentication between clients. Indeed, once two clients have correctly authenticated with a common set of Security Hubs, they will be able to authenticate with each other with the same level of security. DSKE guarantees information-theoretically secure authentication between clients, relying upon the authentication of each client with the Security Hub during the PSRD delivery phase.

DSKE clients, once on-boarded in the system, can send key requests to the Security Hub. These requests can be sent over an insecure channel, i.e. the Internet, and they are processed by the Security Server. If a first client, Alice, is requesting a key with a second client, Bob, the Security Server has to build key instructions consuming some of the PSRD in each Alice and Bob private PSRD tables. These computations must be done locally in order to guarantee information theoretical security for DSKE, and for this reason PSRD with clients must be maintained in a single location within the

Security Hub, i.e. the Security Server. Once the key instructions are computed, these can be delivered to Alice and Bob over an insecure channel.

The internal structure of the Security Hub allows for a very high level of scalability of the system. This scalability is mainly due to the simplification of the logistic operations associated with the PSRD delivery. In a DSKE system with a single-entity Security Hub, clients from any region of the world shall receive the PSRD from a single location (i.e. the location of the Security Hub). As the cost and the risk associated with the PSRD delivery is proportional to the distance between a single-entity Security Hub and the client, costs and risk can be largely reduced with the introduction of the Local Distributors. In this scenario, the large distance between a Security Server and its clients are only covered once, when the Local Distributor for that specific region is set up, and unallocated (aggregate) PSRD is delivered from the Local Distributor to the Security Server. PSRD delivery to local clients happens over way shorter distances, provided that the client is close to any Local Distributor.

9. DSKE Client

A client joins the DSKE system by sending a request to each Security Hub. Each Security Hub, after associating a unique DSKE client ID to the new client, delivers to the DSKE client some PSRD and, at the same time, performs authentication of the client.

The multiple Security Hubs in the system are separately operated entities. When the first client, Alice, wants to establish a key with a second client, Bob, she sends a key request to each Security Hub over a public channel. Key requests contain information about Bob’s ID, the size of the key requested, and other information needed to produce the key at Bob’s side. These key requests are sent independently to each Security Hub, which computes key instructions based on the PSRD they have in common with Alice and Bob. The key instructions are then sent to Bob via a public channel. For each Security Hub, Bob uses its PSRD and the key instruction to build a key share. Each key share is known to only Alice, Bob, and one Security Hub. Alice and Bob implement a (k, n)-secret sharing scheme to build a final key, where k is the number of shares necessary to build the final key, and n is the total number of Security Hubs that Alice chooses to use. Alice and Bob can choose k and n, as well as the set of n Security Hubs that are to be used, dynamically.

Note that, in a similar way, the DSKE system allows a client to request a multi-recipient key, i.e. a key that is to be established with two or more other clients.

10. Secure Application Entity (SAE)

A DSKE key is usually consumed by another entity, for example, a Secure Application Entity (SAE) as specified in the standard ETSI GS QKD 014, which can use the key to secure any of a variety of communications. For example, DSKE keys can be integrated in IPSec via RFC 8784, in TLS via RFC 4279, and can be integrated in other applications that allow for the specification of a pre-shared key. The key is usually employed for encryption, authentication, or both.

In a typical DSKE deployment, a DSKE client is interfaced to a SAE via a secure link, either because the DSKE client is co-hosted in the same device with the SAE, or it is connected to the SAE over a secure Local Area Network (LAN). For example, for site-to-site security, a DSKE client can be installed on a device co-located with one or more network appliances, and the network appliances can send key requests to the DSKE client via a standard API interface such as ETSI GS QKD 014. On a mobile device, the DSKE client can be directly run on the device itself, and expose an API for other applications to request DSKE keys.

11. PSRD Generation, Delivery and Storage

A general deployment of a Security Hub consists of a single Security Server, which can be operated from the cloud or a private colocation, and multiple Local Distributors, that can be operated in different locations depending on the use case. Each Local Distributor can be operated fully independently of the others; it depends on only its associated Security Server.

Whenever a new Local Distributor is created, some amount of unallocated PSRD is generated, and a copy is shipped to the Security Server. This is the first PSRD delivery that must be completed by the Local Distributor in order to become operational. The generation of the PSRD is performed by the Local Distributor, and can happen in different ways. Usually, either a Pseudo-Random Number Generator (PRNG) is employed, a Hardware Random Number Generator (HRNG), a Quantum Random Number Generator (QRNG), or a combination. The difference between the three methods lies in the source of randomness. PRNG are chaotic processes with a very long period, but they require an initial seed. The seed is usually taken from a HRNG, leveraging some physical processes of the machine where the PRNG is hosted. QRNGs, on the other hand, leverage quantum-mechanical processes to generate randomness, and they usually come with a theoretical security proof about the quality of the outcome they produced. The DSKE protocol is agnostic to the type of random number generator used for the PSRD; its security does however depend on its entropy.

Once a Local Distributor is operational, it can fulfill a DSKE client's requests to deliver PSRD, i.e. a portion of the unallocated pool is allocated to that DSKE client, shipped to it and deleted locally. There are, generally speaking, two types of PSRD delivery: unallocated PSRD from a Local Distributor to its Security Server; and allocated PSRD from a Local Distributor to a DSKE Client.

11.1. PSRD Delivery from Local Distributor to Security Server

A typical PSRD delivery between a Local Distributor and a Security Server can involve a large amount of unallocated PSRD (gigabytes, terabytes, or more). A single shipment can provide enough PSRD for several years of operations of a Local Distributor, depending on demand. However, organization-specific security policies may require unallocated PSRD to be deleted and generated afresh regularly, e.g. every six months.

The following is a list of some possible methods for moving the PSRD from a Local Distributor to its Security Server:

Physical delivery using Hardware Security Modules (HSMs). Off-the-shelf devices, for example encrypted USB keys, can be used as well.

In case the Security Server is operated in the cloud, the cloud service provider may offer a service to upload data by physical delivery.

11.2. PSRD Delivery from Local Distributor to DSKE Clients

Note that each use case may place specific requirements on PSRD delivery. However, the DSKE protocol is agnostic to the method used for PSRD delivery. This means that two DSKE Clients can receive PSRD in two different ways, and they would still be able to establish keys over DSKE. The same Local Distributor can also distribute PSRD using different methods for different DSKE Clients.

The following is a list of possible methods for the PSRD distribution for Site-to-Site security, i.e. for the delivery from Security Hub to Quantum Bridge KME:

Physical delivery using secure Hardware Security Modules (HSMs). Off-the shelf devices, for example Kingston encrypted USB keys, can be used.

For military purposes, specialized key fillers can be utilized, for example the KYK-13.

In case the client is operated in the cloud, the cloud service provider may offer a service to upload data by physical delivery on secure hardware that is owned by the cloud service provider.

Quantum Key Distribution: note that, because the number of Local Distributors for a Security Hub is not limited and these may be placed in locations convenient to QKD, the latter's distance limitations do not constrain the deployment of DSKE. QKD can be effectively deployed within DSKE.

The following is a list of possible methods for the PSRD distribution for endpoint security, i.e. for the delivery from Local Distributor to portable devices such as tablets and mobile phones:

QR codes: The Local Distributors are equipped with a feature to print small amounts (approximately 1 kB) of PSRD as QR codes when allocated to a client. This small amount can be used to initially encrypt (using AES-256) and transfer a larger amount of PSRD over the Internet. In a large organization, one could position Local Distributors in the IT room and ask personnel to import PSRD by scanning QR codes. Content secrecy of, and hence access to, these QR codes must therefore be strictly controlled.

SIM or eSIM cards for mobile phones usually come with pre-installed pre-shared keys with the service provider. These keys can be used to encrypt and deliver a larger amount of PSRD to the mobile phone.

NFC (Near Field Communication).

Encrypted USB memories or key loaders can also be used for portable devices like mobile phones or laptops.

11.3. PSRD Storage

The security of PSRD is critical to the security of the DSKE protocol. Every implementation of a DSKE component (including Security Server, Local Distributor, DSKE client, or PSRD transfer mechanism) MUST manage the secrecy and number of copies of the PSRD to meet the system security requirements for the context. Though such requirements are outside the scope of this document, some idealized considerations are listed here:

Only two copies: After generation of the random data to be used as PSRD, two and only copies should exist, and each destroyed when used. This implies that upon transfer of a copy to a new medium (e.g. loading into RAM from disk, or transferring it to an encrypted form), the source copy must be securely destroyed.

Storage security: The security of the storage should be managed so as to ensure the security level demanded of the context. Particularly, security of software, system isolation, electromagnetic emanation and physical access should meet the requirements.

Processing security: Processes that access and manipulate PSRD and derived data should be designed to ensure that there is no leakage of PSRD. Although it is generally infeasible to avoid temporary duplication, every process should end with one copy only of the PSRD or data that is derived from it. For example, in moving data from RAM into a register or vice versa, or if data is combined with other data through an operation (e.g. XOR), a copy is created. The process should ensure that the source copy is always overwritten, and never remains for some other process to possibly access it.

Removable storage of PSRD (including forms used for transferring PSRD between DSKE entities) presents a significant security challenge. Ensuring the security of DSKE forces the same principles to be applied. For example, if a QR code is used to transfer PSRD, this QR code must be transferred in a way that prevents copying in transit, or at a minimum ensures that any possibility of having been copied is detected and managed before use; relatedly, the QR code itself must be securely destroyed upon loading at the destination.

12. DSKE Protocol Specification

The DSKE protocol defines a method for generating and distributing symmetric keys across a network of clients in a secure, efficient, and scalable manner. For the purpose of this document, the Security Hub will be treated as a single entity, i.e. there will be no separation between Security Server and Local Distributors. We now discuss each phase of the protocol in detail.

12.1. Client Registration

When a client registers with a Security Hub, the following happens:

A representative of the client that is acceptable to the Security Hub submits an application to the Security Hub to register the client.

The Security Hub determines whether it will accept the client, including whether it has a suitable mechanism for securely delivering initial keys and subsequent PSRD to the correct entity. If not, it aborts the registration.

The Security Hub assigns an ID to the client that is unique between clients at that Security Hub an assignment that may happen at any time prior to this point, and may be independently linked to a real-world identity. There is no link between this ID and IDs allocated by other Security Hubs to this client: the uniqueness requirement applies only within the set of IDs allocated by that Security Hub.

The Security Hub produces the necessary initial keys, and arranges for secure delivery of the initial keys and the allocated ID to the client. Secure delivery includes two necessary factors: a validation of the correct real-world identity of the recipient, and a proof to the recipient of the identity of the sending Security Hub.

The client receives the ID and initial keys, and it transfers them from the form in which they are sent into its internal storage.

Optional: An "I'm here" message may be sent to alert the Security Hub that the client has received the initial key, which may include a proof that it has been correctly received.

Note: The first PSRD shipment may accompany this delivery; however, separating these deliveries improves security, as the interception of both the initial keys and the PSRD by an adversary is made more difficult, and the two deliveries become independent authentication factors in the registration process.

At the end of the registration process with a Security Hub, the DSKE Client will have a unique DSKE ID assigned to it, together with the initial key(s) shared with the Security Hub.

12.2. PSRD Generation and Distribution

The SH determines that a validated client is to receive additional PSRD, through a request, schedule, tracking of PSRD exhaustion, or other means.

The SH locally allocates the required quantity of unallocated PSRD to the client, determines the PSRD index range within its unallocated PSRD store to package and ship to the client.

The SH transfers the indexed PSRD to a secure shipment for the client, encrypted and authenticated using the PSRD encryption protocol.

SH securely ships the PSRD to the client; each side uses a secure mechanism to validate the real-world identity of the other party, coupled to the delivery. The actual mechanism used is out of scope here.

12.3. Peer Identity Establishment

Peer identities in the form of their DSKE Client IDs as assigned by each Security Hub must be established between DSKE clients, validated against their real-world identities. The system may be designed to build on trust placed by a client in the identity validation performed by each of the Security Hubs.

The mechanism employed for this purpose is out scope for this document.

12.4. Key Establishment

Refer to Appendix A for protocol specifics relating to key establishment.

The sending DSKE client (Alice) requesting the key establishment:

determines with whom a key is to be established, and associated security parameters that will be acceptable to both sender and recipients for the establishment of the specific secret. This includes k, n, the identities of n SHs that both it and the intended recipient(s) are all registered with. The choices are determined by security constraints that the clients require.

determines the secret ID, which is unique for the sender (some relaxation of this requirement is possible, but not used here).

determines the length of the secret.

selects whether it will be:

- (a) DSKE secret transmission: a provided secret, or
- (b) DSKE key establishment: a random secret.

selects and reads the needed amount of PSRD from each of the n SHs, deleting it from its PSRD store.

DSKE client generates shares (with parameters k, n). This may be, for example, in a Shamir scheme by using PSRD from the n SHs. The PSRD from the first (a) k − 1 or (b) k SHs respectively is unaltered as shares of the secret, and the remaining n − k + 1 or n − k respectively shares are derived.

generates a message to each of the SHs. Each message contains:

- The encrypted share (which will be zeros for the first k shares if the optional share randomization is not used)
- Authentication tag for the secret and critical associated data
- PSRD index range (metadata describing the range of the PSRD used in the operation, start index + length with enough PSRD to encrypt the shares and to produce the secret tag and message tag)
- The coordinate xi used in the share for the Security Hub
- The parameters (k, n)
- The secret ID
- Additional Associated Data: handed to the DSKE client by the application layer [not encrypted nor hidden to the SH]
- Sender’s DSKE client ID [TBD: whether this is inferred or explicitly included in the message], as assigned by the SH that the share will be sent through
- Vector of DSKE client IDs for which the key is intended, as assigned by the SH that the share will be sent through.

Sign the entire foregoing using the DSKE message MAC algorithm, using PSRD from the SH, appended to the package

Wrap the message using an AES key with an AEAD mode

key request package = (package, Alice DSKE ID (inferred or explicit), a nonce that was used in the AEAD mode, and an identifier for its key)

Key request package is sent to SH

May receive and validate the SH response; this is not part of the DSKE protocol, but may be desirable for status management depending on the context

The sender may separately alert the recipient(s) to expect the key; this is not part of the DSKE protocol, but may smooth operation.

Security Hub key instruction generation (after receiving a request from Alice):

Verify the secure transport authentication, and decrypt as needed

Verify the DSKE MAC (using PSRD) in the inner package

Decrypt the share using the PSRD and the (start index + length)

Store the share

The SH may send an acknowledgement back to Alice. This is not part of the DSKE protocol, but may facilitate operation.

Each DSKE client receiving the established key:

[TBD: two variants may be considered: an offline-compatible variant in which the SH controls the PSRD indices for the key instructions, and an online-only variant in which each recipient controls the PSRD indices.]

Retrieval of key instructions:

- Offline variant: DSKE client retrieves available or requested specific wrapped key instructions from the required SHs. How the availability thereof is determined (e.g. batch retrieval, or notification by the sender or the SH) is not specified here.

- Online variant: Bob prepares a message for each SH. Each message includes:

- o Sender DSKE client ID and secret IDs that Bob wants

  o Message is then wrapped with AEAD mode under the key for use in this mode, which includes adding any nonce and key identifier, and sent

  o Unwrap the retrieved AEAD messages with the AEAD key.

  o For each message, check the message tag (MAC) using selected PSRD (see Appendix A for detail)

  o Attempt recovery of the secret (see Appendix A for detail).

12.5. Key Validation

The key validation phase of the DSKE protocol is through validation of the DSKE secret tag in Section 3.4.

12.6. Error Handling and Security Considerations

The following transport mechanisms may be used for transmission error management without impacting the security of the DSKE protocol (assuming an authentication layer with the required properties):

- TCP-like retries and delivery
- Error detection and correction protocols
- Cryptographic authentication and encryption protocols

The following is premised on these assumptions:

- The sending and recipient(s) are honest, without security compromise, and functioning correctly;
- The number of the Security Hubs that the receiver will accept key instructions from is not below the minimum k_min that is placed by the receiver on the threshold k of the secret sharing scheme.
- The size of the predetermined set of Security Hubs that the receiver will accept key instructions from is not above the maximum nmax that is placed by the receiver on the number n of shares the secret sharing scheme.
- The protocol is secure against transmission errors, in the sense that these will be detected and discarded by the recipient of the messages, being either a Security Hub or a recipient DSKE client.
- The protocol is secure against re-sending of messages once these have been received, which might have been expected to result in the same secret being reconstructed by a recipient for a second time. It does not inherently prevent the protocol from completing after a delay as a result of delaying or re-ordering the messages, though this can be controlled (if desired) through other mechanisms such as including an expiry time.
- The protocol is secure against compromise, in the sense that an adversary with control of all communication links (excluding initial identity validation and PSRD delivery) will not be able to:

- obtain any information about the secret, other than its length; or

- induce the recipient(s) to reconstruct a secret other than that which is sent by the purported sender.

The DSKE protocol, in itself, is not robust against denial of service attacks that deplete one-time authentication keys, as is used for authenticating the key request and key instruction messages. To protect against this, a wrapping authentication protocol with reusable keys must be employed.

The protocol is robust and will function correctly provided that the number of shares that reach the receiver successfully is not below the threshold k of the secret sharing scheme.

13. Security Model

DSKE's security model is anchored in its resistance to both classical and quantum computational attacks, leveraging the distributed trust established through pre-shared keys and the one-time-pad principle for packet encryption. The model is resilient against known attacks such as eavesdropping, man-in-the-middle, and replay attacks.

Detailed security proofs should be provided to demonstrate the protocol's resistance to various attack vectors (based on https://arxiv.org/abs/2304.13789).

14. Implementation and Integration

The DSKE protocol is designed to be implemented in various network environments, including but not limited to traditional data centers, cloud services, SD-WAN infrastructures, BGP router networks, and others. Its integration is facilitated by its design, which allows it to be adapted for use with a variety of network devices and software platforms.

14.1. Case Studies

IPsec: Illustrating the implementation of DSKE within a VPN environment, demonstrating how DSKE keys are utilized to establish secure communication tunnels, exercising the multiple options for symmetric key use in IPsec.

SD-WAN Deployment: Exploring the benefits of DSKE in an SD-WAN setup, particularly focusing on the economic and operational efficiencies gained by using DSKE over traditional PKI-based systems.

BGP and Routing Security: Discussing the methods for incorporating DSKE into BGP to enhance routing security between autonomous systems.

Cloud Authentication Services: Detailing the process of integrating DSKE into cloud-based authentication services to provide robust security without the overhead associated with PKI.

15. Conclusion

The DSKE protocol represents a significant step forward in the field of practical cryptographic key establishment. It addresses the current and future challenges faced by network security professionals, particularly in the face of the quantum computing threat. By providing a secure, efficient, and scalable method for symmetric key establishment, DSKE can advance as a standard in the space of symmetric key establishment.

16. References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", RFC 2119, March 1997, <https://www.rfc-editor.org/rfc/rfc2119.txt>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", RFC 8174, May 2017, <https://www.rfc-editor.org/rfc/rfc8174.txt>.

[RFC8784] Tschofenig, H., "Using Pre-Shared Keys in the Context of TLS", RFC 8784, May 2020, <https://www.rfc-editor.org/rfc/rfc8784.txt>.

[RFC4279] Eronen, P., "Pre-Shared Key Ciphersuites for Transport Layer Security (TLS)", RFC 4279, December 2005, <https://www.rfc-editor.org/rfc/rfc4279.txt>.

[NIST.SP.800-38D] Dworkin, M., "Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC", NIST SP 800-38D, 2007, <https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-38d.pdf>.

[RFC8452] Gueron, S., "AES-GCM-SIV: Nonce Misuse-Resistant Authenticated Encryption", RFC 8452, September 2019, <https://www.rfc-editor.org/rfc/rfc8452.txt>.

[RFC5996] Kaufman, C., "Internet Key Exchange Protocol Version 2 (IKEv2)", RFC 5996, September 2010, <https://www.rfc-editor.org/rfc/rfc5996.txt>.

[arxiv2304.13789] Lin, J., von Willich, M., and H. Lo, "Composable Security of Distributed Symmetric Key Exchange Protocol", arXiv 2304.13789, 2023, <https://arxiv.org/abs/2304.13789>.

[FOCS.1994] Shor, P.W., "Algorithms for quantum computation: Discrete logarithms and factoring", IEEE Comput. Soc. Press 35th Annual Symposium on Foundations of Computer Science, 1994, <https://doi.org/10.1109/SFCS.1994.365700>.

[NIST.PQC.2016] Chen, L., "Report on post-quantum cryptography", NIST Report on Post-Quantum Cryptography, 2016, <https://csrc.nist.gov/publications/detail/nistir/8105/final>.

[SENSOR.2003] Chan, H., Perrig, A., and D. Song, "Random key predistribution schemes for sensor networks", IEEE Symposium on Security and Privacy 2003, 2003, <https://doi.org/10.1109/SECPRI.2003.1199337>.

Appendix A. Appendix

This appendix provides detail of specific choices of secret sharing scheme and authentication function for generating the secret tag, with the purpose of allowing prototyping the core key establishment protocol of section 3.4. It does not cover management of the PSRD distribution and of managing the AEAD key used in the message wrapper. The protocol is varied slightly from that in Appendix A of reference [1].

In this appendix, ’word’ means a 128-bit quantity, and ’byte’ means an 8-bit quantity. Concatenating (symbol ’||’) 16 bytes produces a word. Concatenation places the first quantity in the least-significant position. Thus, hex_10 || hex_32 || hex_54 || hex_76 = hex_67452310. The prefix ’hex_’ denotes hexadecimal (i.e. base 16), and the prefix ’F_’ denotes an element of the field F in a polynomial basis expressed in hexadecimal. Intermediary zeros are abbreviated ’…’. An underscore ’_’ is used as a digit-grouping separator. The caret symbol (‘^’) indicates the operation of taking a power, and indexing is indicated in trailing brackets (‘[...]’). Where the index is A, it indicates that the variable relates to Alice, and B relates to Bob.

We use the field F = GF(2^128) expressed in a polynomial basis modulo the irreducible polynomial 1 + x + x^2 + x^7 + x^128.

The LSB of a word holds the coefficient of the lowest-degree term (x^0). Thus, the polynomial 1 + x^6 + x^7 + x^126 as an element of F as

F_40000000_00000000_00000000_000000C1 or F_40…0C1

Addition is standard exclusive-or (F_40…0C + F_60…0A = F_20…06).

Multiplication is as polynomials modulo the given irreducible polynomial (identity element F_0…01; F_80…0 × F_0…02 = F_0…087)

Message hash function: h(c,d,y[1],…,y[m]) := d + ⊠1=1..m c^i y[i]

Secret hash function: h(c,d,e,y[1],…,y[m])

Note that these are the same variadic function, with either two or three leading parameters being considered to be the hashing key. Operations (addition, summation, multiplication, integer power) are all in the field F.

k-of-n secret sharing scheme: Shamir secret sharing scheme applied to each word in the sequence in turn, using the same nonzero ’coordinate’ x[i] for every word of a given share. That is, given the corresponding word y[i] of k of the n shares, we can solve for the k variables c[i] in the k simultaneous linear equations y[i] = ⊠j=0,...,k−1 c[j] x[i]^j, and derive the remaining y[i] by using the equations directly. Alternatively, apply this process at the byte level over GF(2^8) with irreducible polynomial 1 + x + x^3 + x^4 + x^8.

Alice

Assume key establishment is required (transferring a secret is similar).

Given: m (secret length in words), n, k, list of recipient SH+receiver client ID pairs, plus a distinct nonzero word "coordinate" per SH (may be assigned per protocol execution, e.g. the share index i, interpreted as an element of F).

Read 2 + 3 + m (128-bit) words of PSRD from each of the n PSRD tables associated with a SH. Retain the first two words as the key for the massage authentication.

Select k of the n SHs (e.g. randomly). For each of these k SHs, the next 3 + m words of these k strings of 3 + m words become the unencrypted share R[A][i] of the secret Y[A][0]. Derive the secret from these shares using the Shamir sharing scheme (the y[0] = c[0] for the c[i]) that, given the k pairs (x[i], y[i]), solve the simultaneous equations y[i] = ⊠ j=0..k−1 c[j] x[i]^j, all operations being in F). Find the y[i] for the remaining n − k SHs.

Divide the words of the secret, in order, are c, d, e, y1, ⋯, ym used calculating secret authentication tag σ [A] = h(c,d,e,y[1],⋯,y[m]). Adjust an input share for the word d so that σ [A] = 0.

[Note: Consider concatenating an encoding of k, n, m, K[A], AB, BA* to the data that is authenticated; these IDs are not transmitted. Here, Bob* assigns an ID AB to Alice that is unique among the IDs that he assigns to other DSKE clients, and vice versa for BA; for multi-recipient networks, such IDs may need to be globally allocated.]

Message format (TBD: bit-lengths of fields, ordering):

M[A][i] = A[i] || B[i]* || K[A] || j[A][i] || k || n || m || x[i] || Z[A][i]

[The SH identity is implied both by t[A][i] and the AEAD wrapping (P[i], A[i]) for every i for which the recipient has established this for Alice’s identity, and similarly for any DSKE client’s identity.

Note that there is no feedback in the reverse direction in the protocol. This is left to higher-level protocols.

Authors’ Addresses

Mattia Montagna
Quantum Bridge Technologies Inc.
108 College Street
Toronto M5G0C6
Canada
Phone: +1-249-225-5099
Email: mattia.montagna@qubridge.io

Manfred von Willich
Quantum Bridge Technologies Inc.
108 College Street
Toronto M5G0C6
Canada
Phone: +1-249-225-5099
Email: manfred.vonwillich@qubridge.io

Melchior Aelmans
Quantum Bridge Technologies Inc.
108 College Street
Toronto M5G0C6
Canada
Phone: +1-249-225-5099
Email: melchior@qubridge.io

Gert Grammel
Juniper Networks
1133 Innovation Way
Sunnyvale, 94098
United States
Phone: +1 888-JUNIPER
Email: ggrammel@juniper.net