Status and Issues for the “Client-Server” Suite of Drafts

draft-ietf-netconf-crypto-types-22
draft-ietf-netconf-trust-anchors-17
draft-ietf-netconf-keystore-24
draft-ietf-netconf-tcp-client-server-12
draft-ietf-netconf-ssh-client-server-27
Draft-ietf-netconf-tls-client-server-27
Draft-ietf-netconf-http-client-server-09
Draft-ietf-netconf-netconf-client-server-25
Draft-ietf-netconf-restconf-client-server-25

NETCONF WG
IETF 113 (Hybrid)
Since IETF 110 (last time presented)

crypto-types:
- Accommodated SecDir review by Valery Smyslov.
- Added "hidden-keys" feature.

trust-anchors:
- Added prefixes to 'path' statements per trust-anchors/issues/1
- Renamed feature "truststore-supported" to "central-truststore-supported".
- Removed two unnecessary/unwanted "min-elements 1" and associated "presence" statements.
- Added Informative reference to "draft-ma-netmod-with-system".

keystore:
- Added prefixes to 'path' statements per trust-anchors/issues/1
- Renamed feature "keystore-supported" to "central-keystore-supported".
- Added features "asymmetric-keys" and "symmetric-keys".
- Added Informative reference to "draft-ma-netmod-with-system".

IEEE 802.1 Liaison
Since IETF 110 (cont.)

tcp-client-server:
- Removed the "tcp-connection-grouping" grouping (now models use the "tcp-common-grouping" directly).
- Added Security Considerations section for the "local-binding-supported" feature.

ssh-client-server:
- Removed the 'supported-authentication-methods' from {grouping ssh-server-grouping}/client-authentication.
- Moved algorithms in ietf-ssh-common (plus more) to IANA-maintained modules
- Added "config false" lists for algorithms supported by the server.
- Added ietf-ssh-common:generate-public-key() RPC for discussion.

tls-client-server:
- Moved algorithms in ietf-ssh-common (plus more) to IANA-maintained modules
- Added "config false" lists for algorithms supported by the server.
- Major update to support TLS 1.3
Since IETF 110 (cont.)

http-client-server:
- Nits

netconf-client-server:
- For netconf-client, augmented-in a 'mapping-required' flag into 'client-identity-mappings' only for the SSH transport, and refined-in a 'min-elements 1' only for the TLS transport.

restconf-client-server:
- Removed Appendix A with fully-expanded tree diagrams.
Open Issue: The "generate-public-key" RPC

- About three years ago, the "crypto-types" draft attempted to define actions for generating private keys. We abandoned these action statements when it became not possible to define a set of algorithm-identifiers that span protocol stacks (e.g., SSH and TLS).

- Now both the "ssh-client-server" and "tls-client-server" drafts have their own IANA-maintained algorithm identifiers, and so it becomes possible to define protocol-specific RPC in each draft.

- Update already made in the SSH draft.
SSH RPC

Tree diagram

```
---x generate-public-key {public-key-generation}?
  +---w input
  |    +---w algorithm sshpka:public-key-algorithm-ref
  |    +---w bits? uint16
  |    +---w (private-key-encoding)?
  |         +--:(cleartext)
  |         |    +---w cleartext? empty
  |         |    +--:(encrypt) {ct:private-key-encryption}?
  |         |         +---w encrypt-with
  |         |         |    +---w (encrypted-by-choice)
  |         |         |         +--:(symmetric-key-ref)
  |         |         |         |    {central-keystore-supported,symmetric-keys}?
  |         |         |         |    +---w symmetric-key-ref?
  |         |         |         |    ks:symmetric-key-ref
  |         |         |         +--:(asymmetric-key-ref)
  |         |         |         |    {central-keystore-supported,asymmetric-keys}?
  |         |         |         |    +---w asymmetric-key-ref?
  |         |         |         |    ks:asymmetric-key-ref
  |         |         +--:(hide) {ct:hidden-keys}?
  |         |         +---w hide? empty
  +--ro output
  +--ro public-key-format identityref
  +--ro public-key binary
  +--ro private-key-format? identityref
  +--ro (private-key-type)
  |    +--:(cleartext-private-key)
  |         +--ro cleartext-private-key? binary
  |    +--:(hidden-private-key) {hidden-keys}?
  |         +--ro hidden-private-key? empty
  |    +--:(encrypted-private-key) {private-key-encryption}?
  |         +--ro encrypted-private-key
  |         +--ro encrypted-by
  |         +--ro encrypted-value-format identityref
  +--ro encrypted-value binary
```
But what about generating TLS keys?

The TLS algorithm registry only defines "cipher suites"
• Not a standalone private key algorithm (like SSH)
• A cipher-suite is a combination of the private key algorithm, encryption algorithm, blocking, and padding used (e.g., tls-rsa-with-aes-256-cbc-sha256)

Should we pass the cipher-suite algorithm identifier and assume the server can identify the private key algorithm?
Slide Transition
Pre-Shared Keys for TLS v1.3 vs. v1.2

ietf-tls-{client, server, common}.yang

2022-03-21

Jeff Hartley
Distinguished Engineer
Problem: PSKs are fundamentally different between TLS versions 1.3 and 1.2

1. TLS v1.2 PSK
   1. Client offers version-specific ciphersuites in Client Hello
   2. Server offers version-specific ciphersuites and identity hint (or null) in Server Hello
   3. Client offers identity

2. TLS v1.3
   1. Client offers version-specific ciphersuites and identity in Client Hello
   2. Server offers version-specific ciphersuites and can immediately start sending data ("0-RTT"), or can fail back to a normal handshake
   3. PSKs can also used to resume sessions

Solution:
1. Redefine the existing YANG "psk" for TLS v1.2 only
2. Create a new TLS v1.3-specific psk object
   1. Ensure RFC8446's pre_shared_key (Section 4.2.11) and psk_key_exchange_modes (Section 4.2.9) extensions are satisfied
   2. Any necessary YANG types, identities, etc.
grouping tls-client-grouping {
    container client-identity {
        choice auth-type {
            case tls12-psk {
                if-feature "client-ident-tls12-psk";
                container tls12-psk {
                    uses ks:local-or-keystore-symmetric-key-grouping;
                    leaf id {
                        type string;
                    }
                }
            }
            case tls13-epsk {
                if-feature "client-ident-tls13-epsk";
                container tls13-epsk {
                    uses ks:local-or-keystore-symmetric-key-grouping;
                    leaf external-identity {
                        type string;
                        mandatory true;
                    }
                    leaf hash {
                        type tlscmn:epsk-supported-hash;
                        mandatory true;
                    }
                    leaf context {
                        type string;
                    }
                    leaf target-protocol {
                        type uint16;
                    }
                    leaf target-kdf {
                        type uint16;
                    }
                }
            }
        }
    }
}
container server-authentication {
    leaf tls12-psks {
        if-feature "server-auth-tls12-psk";
        type empty;
    }
    leaf tls13-epsks {
        if-feature "server-auth-tls13-epsk";
        type empty;
    }
} // container server-authentication

Noteworthy changes to ietf-tls-client.yang -- Desc/Ref fields not shown
grouping tls-server-grouping {
    container server-identity {
        choice auth-type {
            case tls12-psk {
                if-feature "server-ident-tls12-psk";
                container tls12-psk {
                    uses ks:local-or-keystore-symmetric-key-grouping;
                    leaf id_hint {
                        type string;
                    }
                }
            }
            case tls13-epsk {
                if-feature "server-ident-tls13-epsk";
                container tls13-epsk {
                    uses ks:local-or-keystore-symmetric-key-grouping;
                    leaf external-identity {
                        type string;
                        mandatory true;
                    }
                    leaf hash {
                        type tlscmn:epsk-supported-hash;
                        mandatory true;
                    }
                    leaf context {
                        type string;
                    }
                    leaf target-protocol {
                        type uint16;
                    }
                    leaf target-kdf {
                        type uint16;
                    }
                }
            }
        }
    }
    container client-authentication {
        leaf tls12-psks {
            if-feature "client-auth-tls12-psk";
            type empty;
        }
        leaf tls13-epsks {
            if-feature "client-auth-tls13-epsk";
            type empty;
        }
    }
} // container server-identity

container client-authentication {
    leaf tls12-psks {
        if-feature "client-auth-tls12-psk";
        type empty;
    }
    leaf tls13-epsks {
        if-feature "client-auth-tls13-epsk";
        type empty;
    }
} // container client-authentication
Noteworthy changes to ietf-tls-common.yang

identity tls12 {
  if-feature "tls12";
  base tls-version-base;
  status "deprecated";
  description "TLS Protocol Version 1.2."
}

identity tls13 {
  if-feature "tls13";
  base tls-version-base;
  description "TLS Protocol Version 1.3."
}

typedef epsk-supported-hash {
  type enumeration {
    enum sha-256 {
      description "The SHA-256 Hash.";
    }
    enum sha-384 {
      description "The SHA-384 Hash.";
    }
  }
  description "As per Section 4.2.11 of RFC 8446, the hash algorithm supported by an instance of an External Pre-Shared Key (EPSK)."
  I-D.ietf-tls-external-psk-importer: Importing External PSKs for TLS
  I-D.ietf-tls-external-psk-guidance: Guidance for External PSK Usage in TLS";
}
Transition Back
Next Steps

• Validate correctness of TLS 1.3 updates?
• Make updates as needed for the IEEE Liaison.
• Resolve the "generate-key" RPC/action issue.

• Done!  (WG Chairs can publish entire-set to AD)

- Work started in 2014