qpigeon Protocol

Definitions

$$\begin{aligned}
PS &: \text{Public key (for signing)}\\
SS &: \text{Secret key (for signing)}\\
PK &: \text{Public key (for KEM)}\\
SK &: \text{Secret key (for KEM)}\\
K &: \text{Symmetric encryption key}\\
N &: \text{List of used nonces}\\
B &: \text{Contact book}
\end{aligned}$$

$\text{Sign}_{SS}(M) = S$ Signs message M using private key SS creating signature S.

$\text{Sign}^{-1}_{PS}(S, M) = \{0, 1\}$ Verifies message M matches signature S using public key PS. Outputs 1 when signature matches.

$\text{KEM}_{PK}(K) = C$ Encrypts the given key K using public key PK.

$\text{KEM}^{-1}_{SK}(C) = K$ Decrypts the given encrypted key C using secret key SK. KEM stands for Key Encapsulation Mechanism.

$\text{Enc}_K(M) = C$ Encrypts the given message using symmetric key K.

$\text{Enc}^{-1}_K(C) = M$ Decrypts the given ciphertext using symmetric key K.

$\text{Now}() = T$ Outputs the current timestamp.

Registration

Bob registers his keys with the server.

$$\text{Bob has} : PS_{\text{Bob}}, SS_{\text{Bob}}, PK_{\text{Bob}}, SK_{\text{Bob}}$$

$$\text{Bob sends to server} : PS_{\text{Bob}}, PK_{\text{Bob}}$$

$$\begin{aligned}
\text{Server calculates} :& \\
M &= \{0,1\}^{128} && \text{Generate signing challenge.}\\
K &= \{0,1\}^{128} && \text{Generate KEM challenge.}\\
C &= \text{KEM}_{PK_{\text{Bob}}}(K) && \text{Encapsulate KEM challenge.}
\end{aligned}$$

$$\text{Server sends to Bob} : M, C$$

$$\begin{aligned}
\text{Bob calculates} :& \\
S &= \text{Sign}_{SS_{\text{Bob}}}(M) && \text{Sign the signing challenge.}\\
K' &= \text{KEM}^{-1}_{SK_{\text{Bob}}}(C) && \text{Decapsulate the KEM challenge.}
\end{aligned}$$

$$\text{Bob sends to Server} : S, K'$$

$$\begin{aligned}
\text{Server calculates} :& \\
S_{\text{Verify}} &= \text{Sign}^{-1}_{PS_{\text{Bob}}}(S, M) && \text{Verify the signature of the signing challenge.}\\
K &= K' && \text{Verify the KEM challenge response is correct.}\\
& && \text{Once verified, Server records Bob's keys.}
\end{aligned}$$

Contact Request and Accept

Bob adds Alice as a contact.

$$\begin{aligned}
\text{Bob has} &: SS_{\text{Bob}}, PS_{\text{Alice}}, T_{\text{Threshold}}, B, N \\
\text{Server has} &: PS_{\text{Bob}}, PS_{\text{Alice}}, T_{\text{Threshold}}, B, N \\
\text{Alice has} &: SS_{\text{Alice}}, PS_{\text{Bob}}, T_{\text{Threshold}}, B, N
\end{aligned}$$

Bob calculates :

$$\begin{aligned}
T &= \text{Now}() && \text{Get current timestamp.} \\
n &= \{0,1\}^{128} \text{ s.t. } (n, PS_{\text{Bob}}) \notin N && \text{Generate nonce.} \\
N &= N \cup \{(n, PS_{\text{Bob}})\} && \text{Add nonce to list.} \\
S &= \text{Sign}_{SS_{\text{Bob}}}(T||n||PS_{\text{Alice}}) && \text{Sign contact request.} \\
B &= B \cup \{(PS_{\text{Alice}}, PS_{\text{Bob}})\} && \text{Mark Alice as able to send messages to Bob.}
\end{aligned}$$

Bob sends to server : $S, T, n, PS_{\text{Alice}}$

Server calculates :

$$\begin{aligned}
S_{\text{Verify}} &= \text{Sign}^{-1}_{PS_{\text{Bob}}}(S, T||n||PS_{\text{Alice}}) && \text{Verify contact request is from Bob.} \\
T &> \text{Now}() - T_{\text{Threshold}} && \text{Verify contact request is recent.} \\
(n, PS_{\text{Bob}}) &\notin N && \text{Verify nonce is new.} \\
N &= N \cup \{(n, PS_{\text{Bob}})\} && \text{Add old nonce to list.} \\
B &= B \cup \{(PS_{\text{Alice}}, PS_{\text{Bob}})\} && \text{Mark Alice as able to send messages to Bob.}
\end{aligned}$$

Server sends to Alice : S, T, n

Alice calculates :

$$\begin{aligned}
S_{\text{Verify}} &= \text{Sign}^{-1}_{PS_{\text{Bob}}}(S, T||n||PS_{\text{Alice}}) && \text{Verify contact request is from Bob.} \\
& && \text{If } S_{\text{Verify}} = 0\text{, reject.} \\
T &> \text{Now}() - T_{\text{Threshold}} && \text{Verify contact request is recent.} \\
(n, PS_{\text{Bob}}) &\notin N && \text{Verify nonce is new.} \\
N &= N \cup \{(n, PS_{\text{Bob}})\} && \text{Add old nonce to list.} \\
B &= B \cup \{(PS_{\text{Alice}}, PS_{\text{Bob}})\} && \text{Mark Alice as able to send messages to Bob.} \\
\\
T &= \text{Now}() && \text{Get current timestamp.} \\
n &= \{0,1\}^{128} \text{ s.t. } (n, PS_{\text{Alice}}) \notin N && \text{Generate nonce.} \\
N &= N \cup \{(n, PS_{\text{Alice}})\} && \text{Add nonce to list.} \\
S &= \text{Sign}_{SS_{\text{Alice}}}(T||n||PS_{\text{Bob}}) && \text{Sign contact request.} \\
B &= B \cup \{(PS_{\text{Bob}}, PS_{\text{Alice}})\} && \text{Mark Bob as able to send messages to Alice.}
\end{aligned}$$

Alice sends to server : S, T, n, PS_{Bob}

Server calculates :

$$\begin{aligned}
S_{\text{Verify}} &= \text{Sign}^{-1}_{PS_{\text{Alice}}}(S, T||n||PS_{\text{Bob}}) && \text{Verify contact request is from Alice.} \\
& && \text{If } S_{\text{Verify}} = 0\text{, reject.} \\
T &> \text{Now}() - T_{\text{Threshold}} && \text{Verify contact request is recent.} \\
(n, PS_{\text{Alice}}) &\notin N && \text{Verify nonce is new.} \\
N &= N \cup \{(n, PS_{\text{Alice}})\} && \text{Add old nonce to list.} \\
B &= B \cup (PS_{\text{Bob}}, PS_{\text{Alice}}) && \text{Mark Bob as able to send messages to Alice.}
\end{aligned}$$

Server sends to Bob : S, T, n

Bob calculates :

$$\begin{aligned}
S_{\text{Verify}} &= \text{Sign}^{-1}_{PS_{\text{Alice}}}(S, T||n||PS_{\text{Bob}}) && \text{Verify contact request is from Alice.} \\
& && \text{If } S_{\text{Verify}} = 0\text{, reject.} \\
T &> \text{Now}() - T_{\text{Threshold}} && \text{Verify contact request is recent.} \\
(n, PS_{\text{Alice}}) &\notin N && \text{Verify nonce is new.} \\
N &= N \cup \{(n, PS_{\text{Alice}})\} && \text{Add old nonce to list.} \\
B &= B \cup (PS_{\text{Bob}}, PS_{\text{Alice}}) && \text{Mark Bob as able to send messages to Alice.}
\end{aligned}$$

Public Key (for KEM) Distribution

Alice sends a public key (for KEM) PK_{Alice} to Bob.

Alice has : $SS_{\text{Alice}}, PK_{\text{Alice}}$
Server has : PS_{Alice}
Bob has : PS_{Alice}

Alice calculates :

$S = \text{Sign}_{SS_{\text{Alice}}}(PK_{\text{Alice}})$ Signs public key.

Alice sends to Server : S, PK_{Alice}

Server calculates :

$S_{\text{Verify}} = \text{Sign}^{-1}_{PS_{\text{Alice}}}(S, PK_{\text{Alice}})$ Verify message is from Alice.

If $S_{\text{Verify}} = 0$, reject.

Server sends to Bob : S, PK_{Alice}

Bob calculates:

$S_{\text{Verify}} = \text{Sign}^{-1}_{PS_{\text{Alice}}}(S, PK_{\text{Alice}})$ Verify message is from Alice.

If $S_{\text{Verify}} = 0$, reject.

Bob sends message to Alice

Bob sends a given message M to Alice.

$$\begin{aligned}
\text{Bob has} &: SS_{\text{Bob}}, PK_{\text{Alice}}, N \\
\text{Server has} &: PS_{\text{Bob}}, T_{\text{Threshold}}, B, N \\
\text{Alice has} &: SK_{\text{Alice}}, PS_{\text{Bob}}, T_{\text{Threshold}}, B, N
\end{aligned}$$

$$\begin{aligned}
&\text{Bob calculates :} \\
K &= \{0,1\}^n && \text{Generates key of length } n. \\
C_K &= \text{KEM}_{PK_{\text{Alice}}}(K) && \text{Encrypts key.} \\
C_M &= \text{Enc}_K(M) && \text{Encrypts message.} \\
T &= \text{Now}() && \text{Get current timestamp.} \\
n &= \{0,1\}^{128} \text{ s.t. } (n, PS_{\text{Bob}}) \notin N && \text{Generate nonce.} \\
N &= N \cup \{(n, PS_{\text{Bob}})\} && \text{Add nonce to list.} \\
S &= \text{Sign}_{SS_{\text{Bob}}}(T||n||C_K||C_M) && \text{Sign message.}
\end{aligned}$$

$$\text{Bob sends to server} : S, T, n, C_K, C_M$$

$$\begin{aligned}
&\text{Server calculates :} \\
S_{\text{Verify}} &= \text{Sign}^{-1}_{PS_{\text{Bob}}}(S, T||n||C_K||C_M) && \text{Verify message is from Bob.} \\
& && \text{If } S_{\text{Verify}} = 0\text{, reject.} \\
T &> \text{Now}() - T_{\text{Threshold}} && \text{Verify message is recent.} \\
(n, PS_{\text{Bob}}) &\notin N && \text{Verify nonce is new.} \\
N &= N \cup \{(n, PS_{\text{Bob}})\} && \text{Add old nonce to list.} \\
(PS_{\text{Bob}}, PS_{\text{Alice}}) &\in B && \text{Verify Bob can message Alice.}
\end{aligned}$$

$$\text{Server sends to Alice} : S, T, n, C_K, C_M$$

$$\begin{aligned}
&\text{Alice calculates :} \\
S_{\text{Verify}} &= \text{Sign}^{-1}_{PS_{\text{Bob}}}(S, T||n||C_K||C_M) && \text{Verify message is from Bob.} \\
& && \text{If } S_{\text{Verify}} = 0\text{, reject.} \\
T &> \text{Now}() - T_{\text{Threshold}} && \text{Verify message is recent.} \\
(n, PS_{\text{Bob}}) &\notin N && \text{Verify nonce is new.} \\
N &= N \cup \{(n, PS_{\text{Bob}})\} && \text{Add old nonce to list.} \\
(PS_{\text{Bob}}, PS_{\text{Alice}}) &\in B && \text{Verify Bob can message Alice.} \\
K &= \text{KEM}^{-1}_{SK_{\text{Alice}}}(C_K) && \text{Decrypt key.} \\
M &= \text{Enc}^{-1}_K(C_M) && \text{Decrypt message.}
\end{aligned}$$