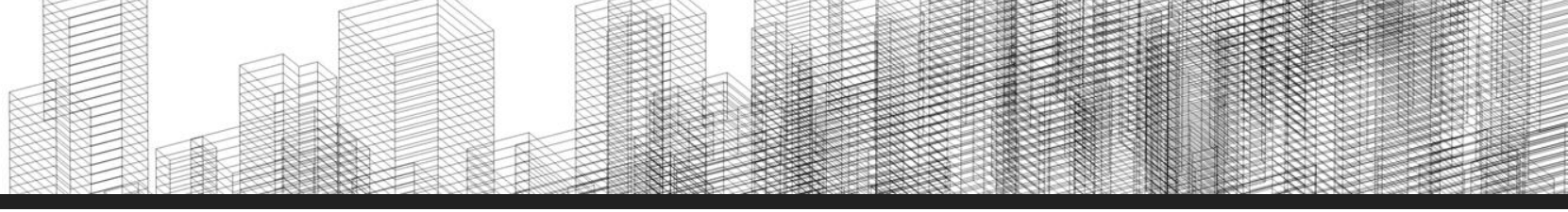
Experiments on modeling and verification of post-quantum protocols using Maude

Víctor García
vicgarv2@upv.es

Santiago Escobar
sescobar@upv.es

Universitat Politècnica de València
VRAIN - Valencian Research Institute for Artificial Intelligence



Contents

- Threat of quantum computers
  - Shor's algorithm
  - Grover's algorithm

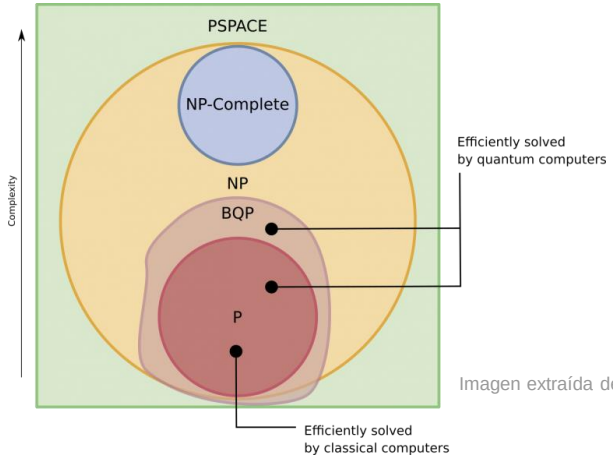
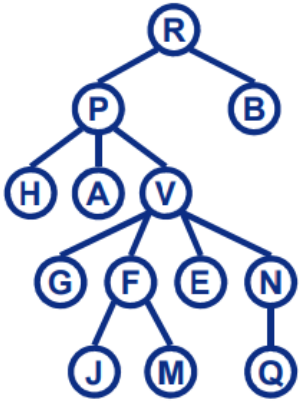


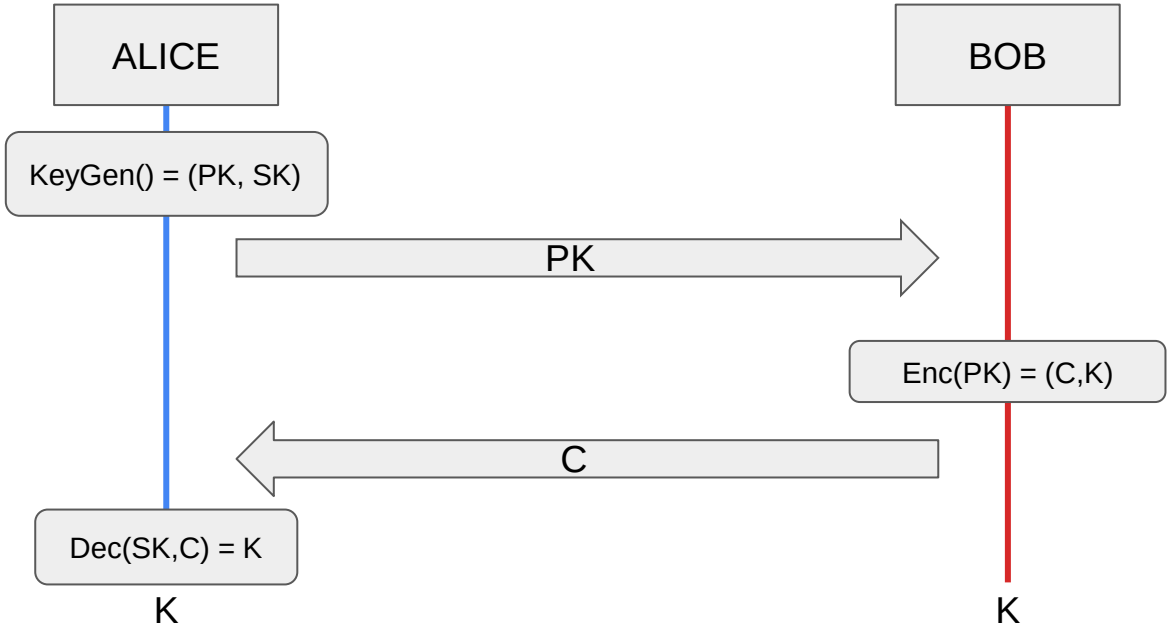
Imagen extraída de https://www.quantum-bits.org/?p=2059

- Solution by the NIST with the Post-Quantum Cryptography project
  - Round 3 (2020)
    - Key Encryption and Establishment: Classic McEliece, Kyber, NTRU, SABER
    - Digital signature: DILITHIUM, FALCON, Rainbow

- Types of analysis
  - Computational
    - Mathematical proofs and probabilities
    - Keys, messages,... are bit strings
    - Closer to reality, used by cryptographer, already applied to Kyber
  - Symbolic
    - Cryptographic primitives as black boxes
    - Keys, messages,… are symbols
    - Suitable for automation and easier to understand for non experts of cryptography

- Maude is a modeling, programming and verification language
- Explicit state model checking using *search* or LTL properties
- Origins at Stanford, California
- Project members
  - USA
  - Norway
  - Spain




ALICE
BOB
KeyGen() = (PK, SK)
PK
Enc(PK) = (C,K)
C
Dec(SK,C) = K
K
K

ALICE

$\texttt{KEM.KeyGen}()$
$z \leftarrow \mathcal{B}^{32}$
$(pk, sk') = \text{CPAPKE.}\texttt{KeyGen}()$
$sk = (sk'||pk||\text{H}(pk)||z)$
return (pk, sk)

$\xrightarrow{pk}$

$\texttt{KEM.Enc}(pk)$
$m_0 \leftarrow \mathcal{B}^{32}$
$m = \text{H}(m_0)$
$(\bar{K}, r) = \text{G}(m||\text{H}(pk))$
$c = \text{CPAPKE.}\texttt{Enc}(pk, m, r)$
$K = \text{KDF}(\bar{K}, \text{H}(c))$
return (c, K)

$\xleftarrow{c}$

$\texttt{KEM.Dec}(c, sk)$
$(\mathbf{s}||pk||\text{H}(pk)||z) = sk$
$m' = \text{CPAPKE.}\texttt{Dec}(c, \mathbf{s})$
$(\bar{K}', r') = \text{G}(m'||\text{H}(pk))$
$c' = \text{CPAPKE.}\texttt{Enc}(pk, m', r')$
if $c = c'$ **then return** $K = \text{KDF}(\bar{K}', \text{H}(c))$
else return $K = \text{KDF}(z, \text{H}(c))$

BOB

Veamos ahora porqué esas primitivas, qué hemos visto a alto nivel, tanto para la generación de claves, encriptación y des-encriptación, son resistentes ante la capacidad de cómputo de un adversario cuántico. La seguirdad de Kyber se basa en la dificultad de resolver el LWE problem sobre modular látices. Bien, Kyber trabaja con vectores y matrices de polinomios con diversas operaciones sobre estos, como pueden ser la concatenación, la trasposición, el producto u otras más complejas como la función hash y la derivación de claves. Lo importante aquí es fijarnos que en el paso de Desencriptación de Alice, esta lo que hace es aplicar la desencripacion para obtener un valor proximo a m que luego usa para volver a encriptar y así comparar lo que ha extraído con lo que ha recibido, que a veces, pero con baja probabilidad pueden no ser iguales. ¿Y a que se debe que sean distintos?.

ALICE

BOB

CPAPKE.KeyGen()

$d \leftarrow \mathcal{B}^{32}$

$(\rho, \sigma) = \mathrm{G}(d)$

$R_q^{k \times k} \ni \mathbf{A} = \mathrm{generate}(\rho)$

$R_q^k \ni \mathbf{s}, \mathbf{e} \leftarrow \mathrm{sampleCBD}(\sigma)$

$\mathbf{t} = \mathbf{A}\mathbf{s} + \mathbf{e}$

$pk = (\mathbf{t}||\rho)$

$sk = \mathbf{s}$

return (pk, sk)

CPAPKE.Dec(c, sk)

$(c_1||c_2) = c$

$\mathbf{u}' = \mathrm{Decompress}_q(c_1, d_u)$

$v' = \mathrm{Decompress}_q(c_2, d_v)$

$m' = \mathrm{Compress}_q(v' - \mathbf{s}^T\mathbf{u}', 1)$

return m'

CPAPKE.Enc(pk, m, r)

$(\mathbf{t}||\rho) = pk$

$R_q^{k \times k} \ni \mathbf{A} = \mathrm{generate}(\rho)$

$R_q^k \ni \mathbf{r}, \mathbf{e}_1 \leftarrow \mathrm{sampleCBD}(r)$

$R_q \ni e_2 \leftarrow \mathrm{sampleCBD}(r)$

$\mathbf{u} = \mathbf{A}^T\mathbf{r} + \mathbf{e}_1$

$v = \mathbf{t}^T\mathbf{r} + e_2 + \mathrm{Decompress}_q(m, 1)$

$c_1 = \mathrm{Compress}_q(\mathbf{u}, d_u)$

$c_2 = \mathrm{Compress}_q(\mathbf{v}, d_v)$

return $c = (c_1||c_2)$

Pues si nos metemos dentro de estas otras operaciones de encriptación y desencriptación podremos ver la aparicion de vectores que son muestreados o generados de forma probabilistíca. Algunos de estos vectores hacen la función de ruido. De esta forma, el uso de ese ruido para construir el criptograma c, hace que sea imposible para un atacante poder obtener el mensaje a compartir entre los participantes. Y gracias a este ruido la complejidad de resolución será NP.

- Build a higher level model of kyber based on the Dolev-Yao intruder approximation
- Present an extended symbolic analysis on the new model
  - Verify its correctnes respect to the original specification
    - Only one key is shared between to participants
    - A participant can never learn the secret key of another participant
    - A key can not be derived from a message of the network
  - Also prove a MITM attack is found under Dolev-Yao's model assumptions

- We are working on a more detailed model than previous works in Maude
- And a more elaborated verification of it
- Final goal:
  - Framework which can be used to model and verify this and other kinds of post-quantum protocols