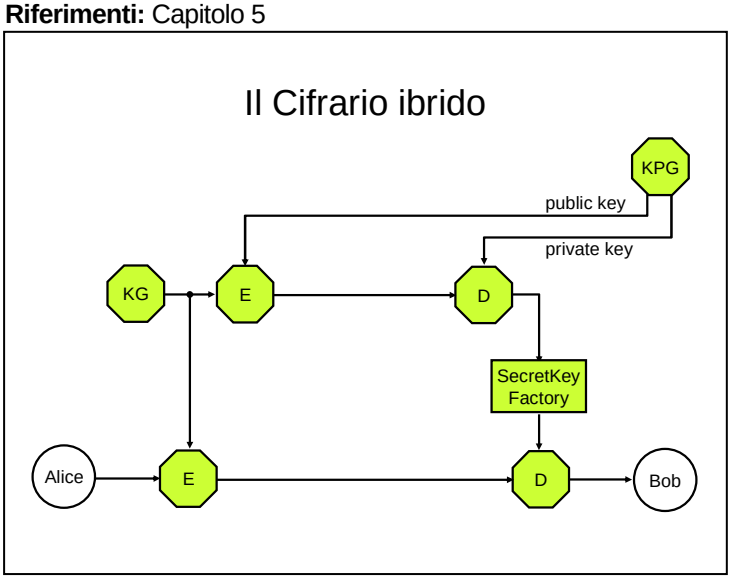
6. Esercitazioni consigliate

6.1 Il Cifrario ibrido

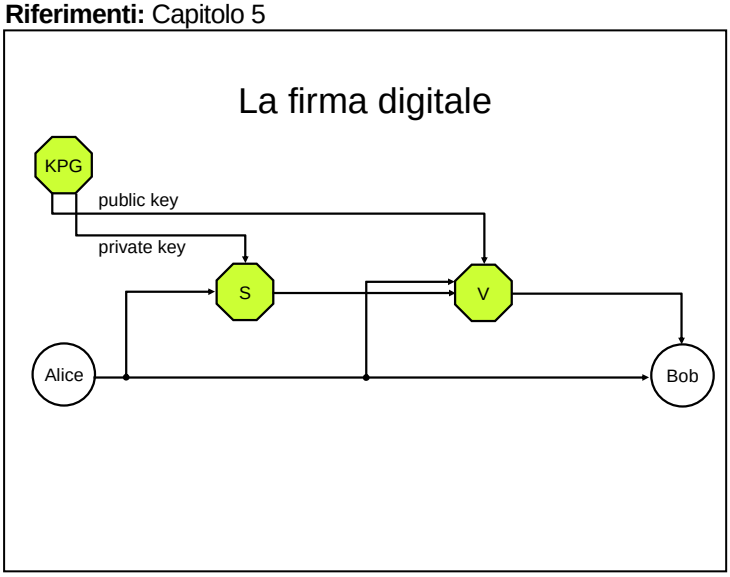
Obiettivo formativo – Capire come un Cifrario asimmetrico possa costituire un'alternativa allo scambio D-H ai fini di consentire la condivisione di una chiave segreta a due utenti che non hanno accordi precedenti.
Riferimenti: Capitolo 5



Esperimenti:
1. Modellare all'interno di una BOX-for il sistema indicato in figura. Configurare la BOX per una sola iterazione.
2. Scegliere e configurare un Cifrario simmetrico ed un Cifrario asimmetrico.
3. Configurare in corrispondenza i blocchi KG e KPG; metterli in esecuzione.
4. Eseguire il Cifrario asimmetrico e verificare che il componente SecretKeyFactory, suggerito quando sono in gioco provider diversi, può anche non essere presente.
5. Mettere in esecuzione il Cifrario simmetrico e verificare che Bob riceve correttamente qualsiasi testo in chiaro generato da Alice.
6. Farsi fornire da S-vLab il codice didattico delle elaborazioni eseguite all'interno della BOX.
7. Trasportare il codice didattico nell'ambiente di sviluppo Eclipse: metterlo in esecuzione dopo averlo ottimizzato e completato.

6.2 La firma digitale

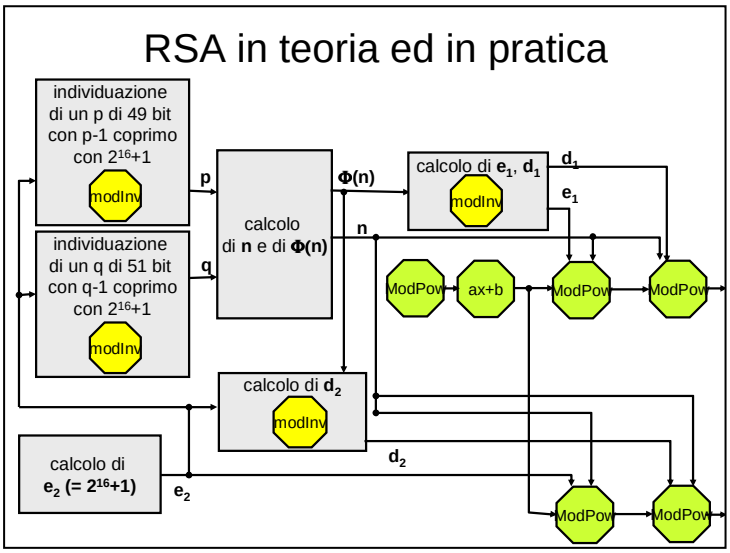
Obiettivo formativo – Sperimentare uno schema di firma con appendice. Prendere atto dei metodi "signature" e "verify" della classe Signature.
Riferimenti: Capitolo 5



Esperimenti:
1. Modellare il sistema di figura.
2. Attraverso l'Esperto, consultare la voce relativa alla classe Signature nella pagina "Standard names - Sun", e verificare che, se la classe viene opportunamente configurata, il calcolo dell'impronta del messaggio è incluso nei metodi "sign()" e "verify()".
3. Configurare i blocchi KPG, S, V per RSA, scegliendo "None" come valore del parametro "mask generation function".
4. Far generare ad Alice un testo "lungo".
5. Eseguire lo schema.
6. Prendere atto della dimensione dell'etichetta e dei tempi di esecuzione dei blocchi S, V.
7. Prendere atto su Bob dell'esito della verifica.
8. Verificare che l'algoritmo di firma configurato in precedenza è deterministico. Imporre "MGF1" come valore per il parametro "mask generation function" dei componenti S e V e verificare che l'algoritmo diventa probabilistico.
9. Ripetere le prove 3-7 con l'algoritmo di firma DSA.

6.3 Generazione di una coppia di chiavi RSA

Obiettivo formativo – Verificare l'algoritmo G di RSA imponendo dapprima un esponente e casuale, e poi l'esponente $2^{16}+1$. Prendere atto della disponibilità del componente: "modInv", che fornisce, quando esiste, l'inverso moltiplicativo di un intero rispetto ad un modulo dato (nel caso in cui l'inverso moltiplicativo non esista, il componente restituisce "false" sull'output "success").
Riferimenti: Capitolo 5



Esperimenti:

1. Modellare con i componenti di S-vLab i blocchi rettangolari indicati in figura e corrispondenti ai passi elementari dell'algoritmo G (generazione di una coppia di chiavi RSA), in modo tale che i valori (p-1) e (q-1) siano coprimi con $2^{16}+1$.
2. Impiegare i valori di n, d_1, e_1 prima individuati per modellare la cifratura e la decifrazione di RSA con il componente ModPow.
3. Generare con un modPow seguito da un componente "ax+b" il BigInteger 2^{99} + 35428 ed impiegarlo come testo in chiaro.
4. Calcolare una seconda coppia e_2, d_2 imponendo $e_2 = 2^{16}+1$, con riferimento allo stesso n precedentemente calcolato. Impiegare i valori di n, d_2, e_2 per modellare la cifratura e la decifrazione di RSA con il componente ModPow, utilizzando lo stesso testo in chiaro precedentemente generato.
5. Confrontare e giustificare i tempi di esecuzione delle due cifrature di RSA.