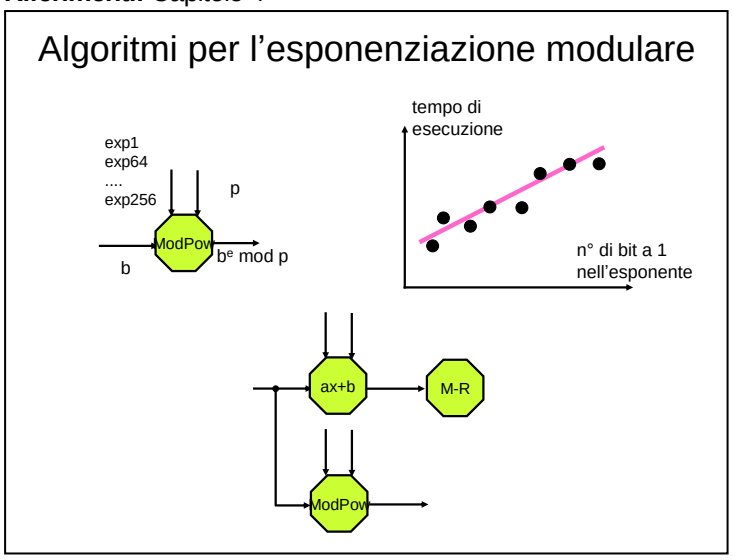
5.a Esercitazioni proposte in laboratorio

5.1 Algoritmi per l'esponenziazione modulare

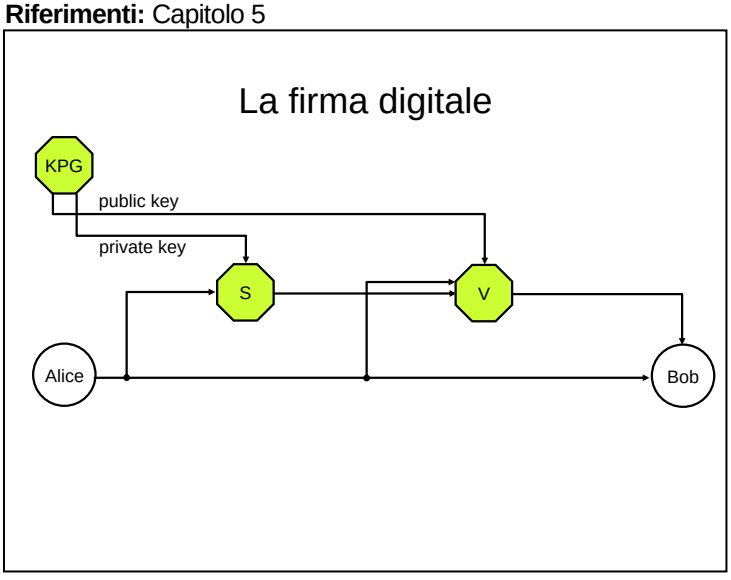
Obiettivo formativo – Verificare alcune proprietà notevoli degli algoritmi per l'esponenziazione modulare. Verificare le proprietà di un *safe prime*.
Riferimenti: Capitolo 4



Esperimenti:
1. Prelevare il file pge.txt dal sito del corso e caricarlo nella cartella "Files" del progetto corrente. Aprire il file, copiare il BigInteger denominato "p" ed incollarlo nel campo *modulus* della scheda In/Out di un componente ModPow. Copiare il BigInteger "g" ed incollarlo nel campo *base*. Copiare il BigInteger "exp1" ed incollarlo nel campo *exponent*. Fare un certo numero di prove ed annotare l'ultimo valore del tempo di esecuzione. Ripetere con "exp64", "exp128", "exp192", "exp256"; costruire un grafico che riporti i risultati ottenuti e giustificarli.
2. Il "p" impiegato in precedenza è un s*afe prime*. Verificarlo sia individuando il corrispondente primo di Sophie Germain (il blocco denominato "ax+b" consente di calcolare 0,5 p – 0,5) , sia controllando che è congruente a 5 mod 6.
3. Ripetere la prova con p=17.

5.2 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

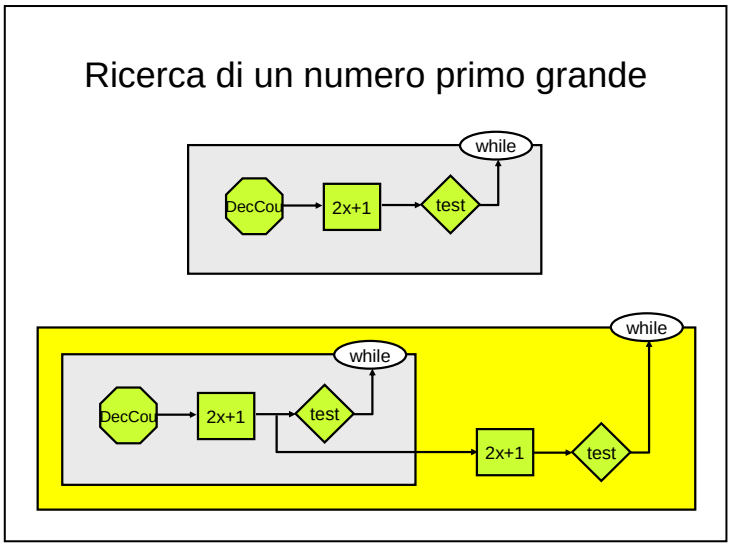


Esperimenti:
1. Modellare il sistema di figura.
2. Attraverso l'Esperto, consultare la voce relativa alla classe Signature nella pagina "Standard names - Sun", e verificare che, se la classe viene opportunamente configurata, il calcolo dell'impronta del messaggio è incluso nei metodi "sign()" e "verify()".
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.

5.b Esercitazioni aggiuntive (facoltative) da svolgere autonomamente

5.3 Generazione di numeri primi

Obiettivo formativo – Prendere atto della distribuzione dei numeri primi. Valutare la probabilità che un numero scelto a caso sia primo. Sperimentare un metodo per la generazione di un safe prime.
Riferimenti: Capitolo 4



Esperimenti:

1. Sperimentare il procedimento per la generazione di un numero primo grande, istanziando all'interno di una BOX-while un componente DecCounter (inizializzato con il valore di 2^28, che può essere calcolato separatamente con un ModPow), un componente "ax+b" (configurato con a=2 e b=1, così da produrre in uscita un numero sicuramente dispari), e un test di Miller-Rabin.
2. Eseguire la BOX più volte, al fine di generare più numeri primi. Individuare, grazie al log prodotto dal mentore, il numero primo ottenuto al termine di ogni ciclo "while". Verificare che i primi così trovati fanno parte della lista dei numeri primi che può essere scaricata dal sito "The Prime Pages", accessibile attraverso l'Esperto.
3. Analizzare la scheda Codice del test di Miller-Rabin. Accedere a Javadoc e prendere atto dei parametri del metodo.
4. Istanziare una BOX-while e collocare al suo interno una seconda BOX-while che contenga un DecCounter, un componente "ax+b" ed il test di Miller-Rabin, collegati e configurati come descritto al punto 1. Collegare l'uscita della box più interna ad un secondo componente "ax+b", configurato con a=2 e b=1, ed un test di Miller-Rabin, così che la box esterna termini la sua esecuzione dopo aver trovato un safe prime.
5. Verificare che il safe prime generato al punto 4 è congruente a 5 modulo 6.