Introduzione alle esercitazioni

Sicurezza dell'Informazione M

A.A. 2011 - 2012

Anna Riccioni

anna.riccioni@unibo.it

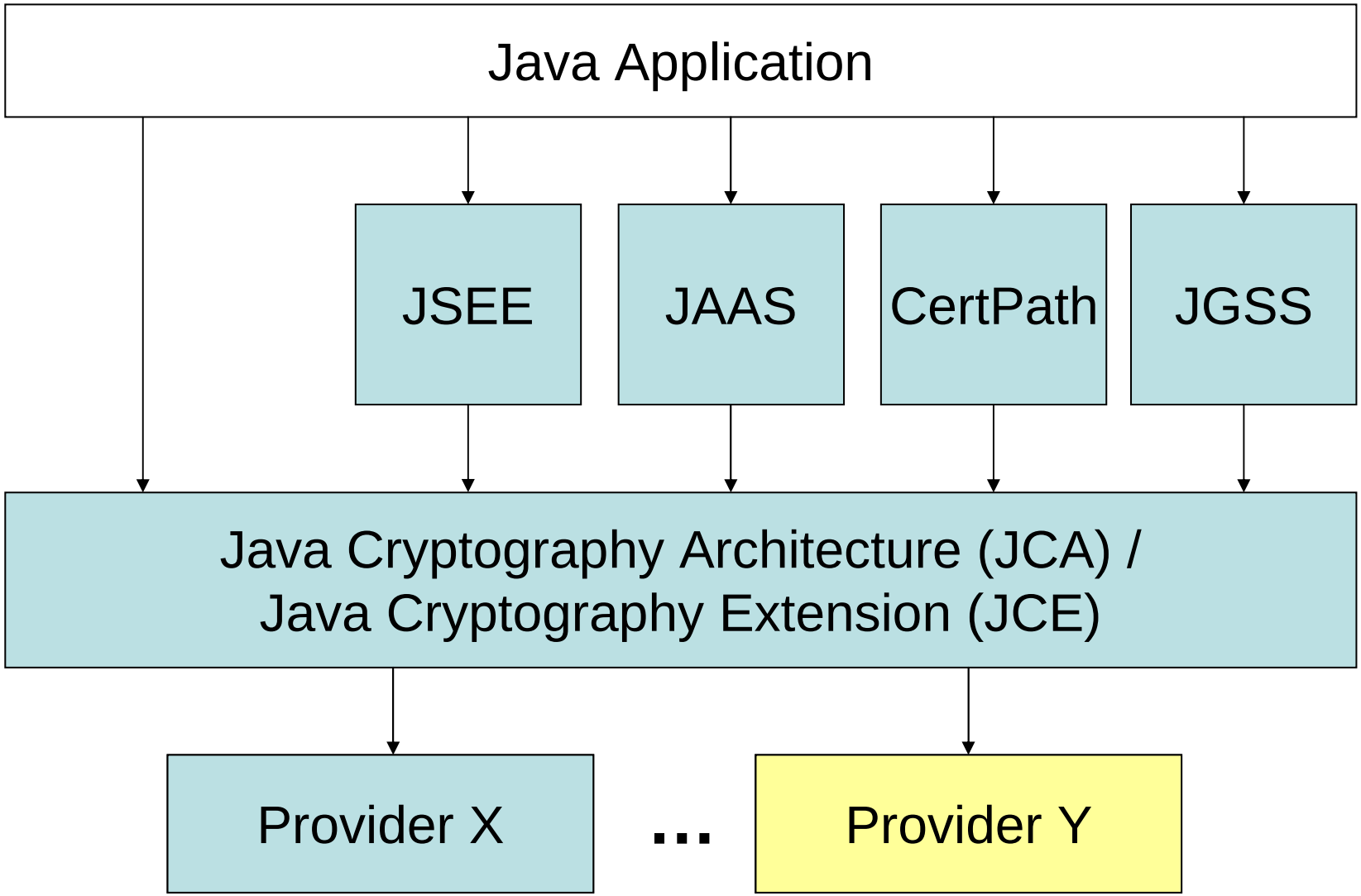
Pagina web del laboratorio di Sicurezza - M

- http://lia.deis.unibo.it/Courses/SicurezzaM1112/ -> Laboratorio Virtuale
  - avvisi e informazioni sulle esercitazioni
  - slide e tracce degli esercizi proposti
  - possibili soluzioni degli esercizi di programmazione
  - materiali necessari per lo svolgimento delle attività suggerite
    - S-vLab: laboratorio virtuale per la sperimentazione, analisi e progettazione di meccanismi, funzioni, servizi e protocolli per la sicurezza dell’informazione

Svolgimento delle esercitazioni

1. Attività sperimentali (S-vLab):
   - verificare le teorie viste a lezione
   - ricostruire ed applicare i modelli studiati
   - simulare il comportamento di sistemi e protocolli per analizzarne robustezza e performance
2. Esercizi di programmazione (Ecplise o altri IDE):
   - Java Security
   - provider crittografici

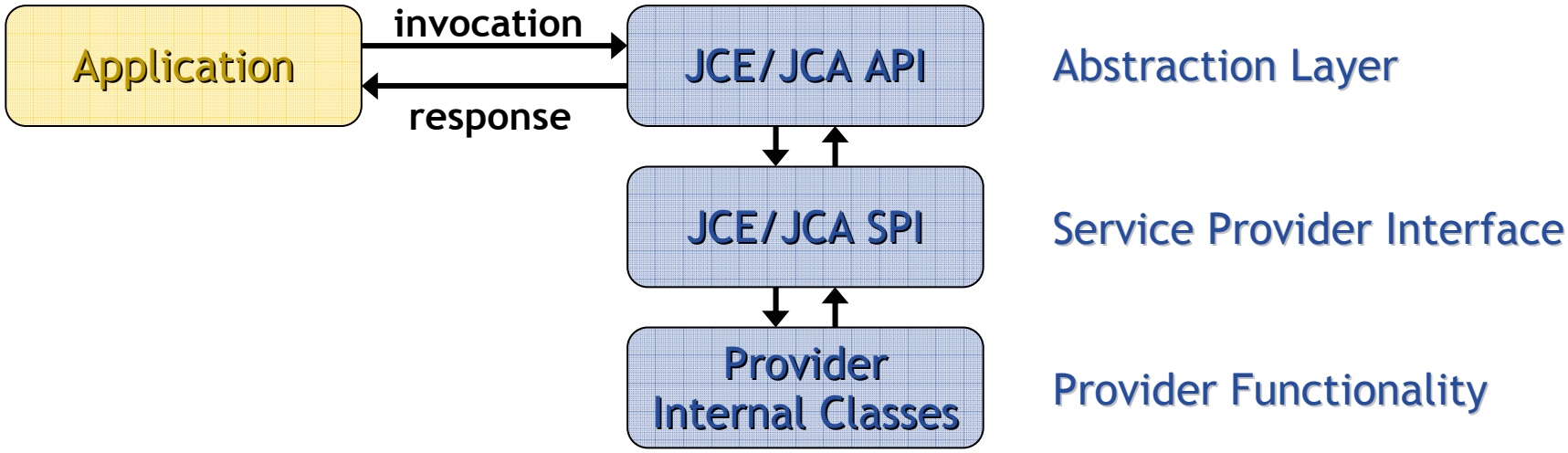
Java Security



Java Security

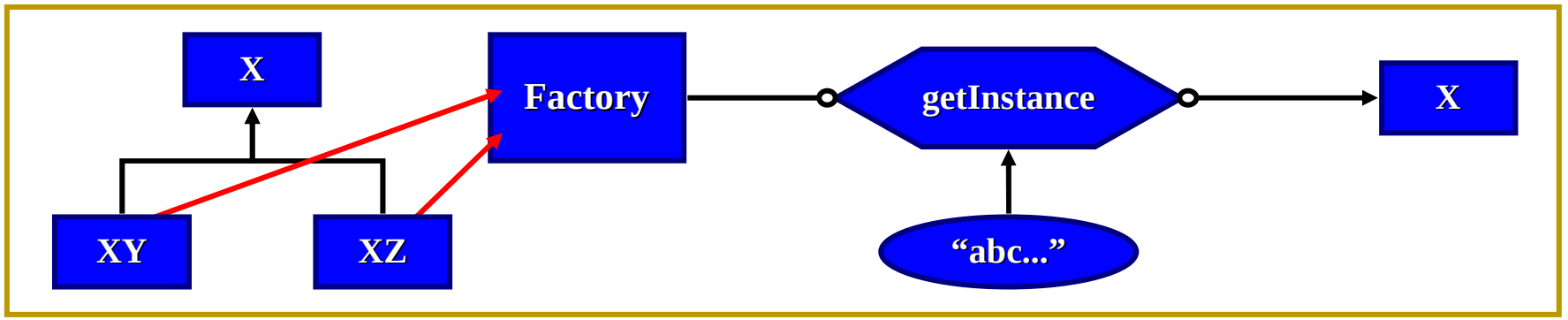
- Java Cryptography Architecture
  - definisce classi astratte per molte funzionalità
  - implementa alcune funzionalità (autenticazione)
- Java Cryptography Extension
  - definisce le API crittografiche complete
  - implementa tutte le altre funzionalità (cifratura / decifrazione)
- Provider-based Architecture
  - classi astratte "engine" che dichiarano le funzionalità di un certo algoritmo crittografico
  - classi concrete "provider" che implementano un insieme di funzionalità entro un Cryptographic Service Provider (CSP)

Provider-based Architecure



- Cryptographic Service Provider (CSP)
  - uno o più package che implementano uno o più servizi crittografici esposti dalla JCA
  - più provider installati, anche se di produttori differenti, possono coesistere

Pattern Factory



- Concretamente:
  - un oggetto viene istanziato non più con la parola chiave `new`, ma grazie al metodo statico `getInstance(String nome)`
- Questo metodo può restituire formalmente delle interfacce, separando completamente:
  - l'uso (espresso dalle interfacce)
  - gli aspetti implementativi (legati alle classi utilizzate internamente).

Provider-based Architecture

- Indipendenza dall'implementazione
  - più provider possono coesistere nello stesso JRE
  - provider diversi possono offrire funzionalità diverse
  - provider diversi possono offrire implementazioni diverse delle stesse funzionalità
- interoperabilità (provider e applicazioni non sono necessariamente strettamente legati)
- estendibilità

Provider-based Architecture

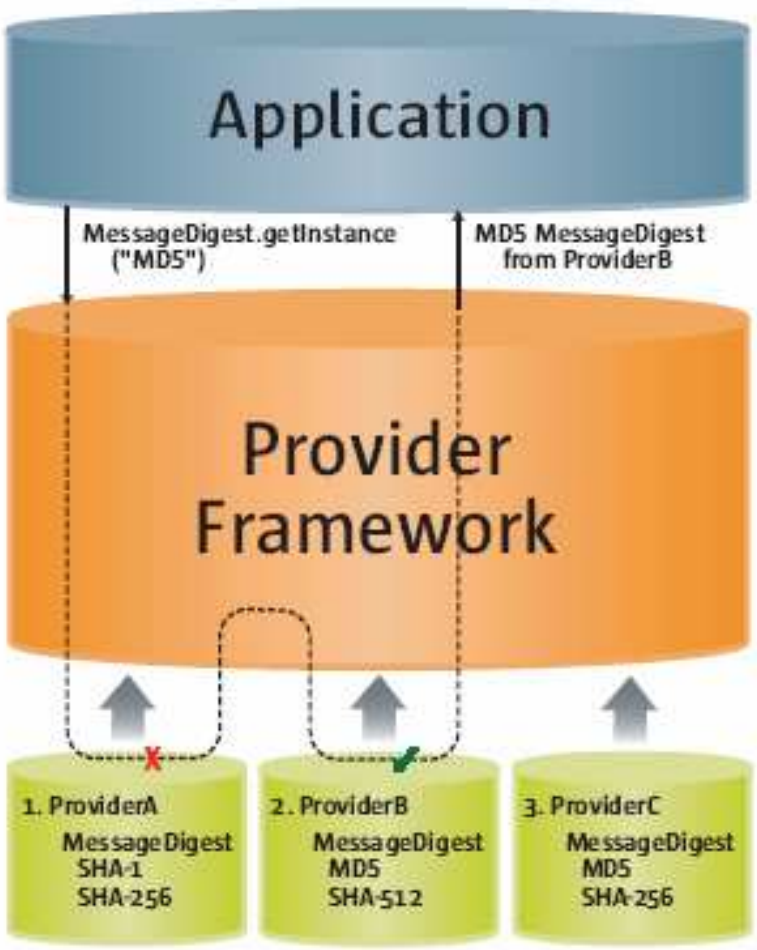


Figure 1 – Provider searching

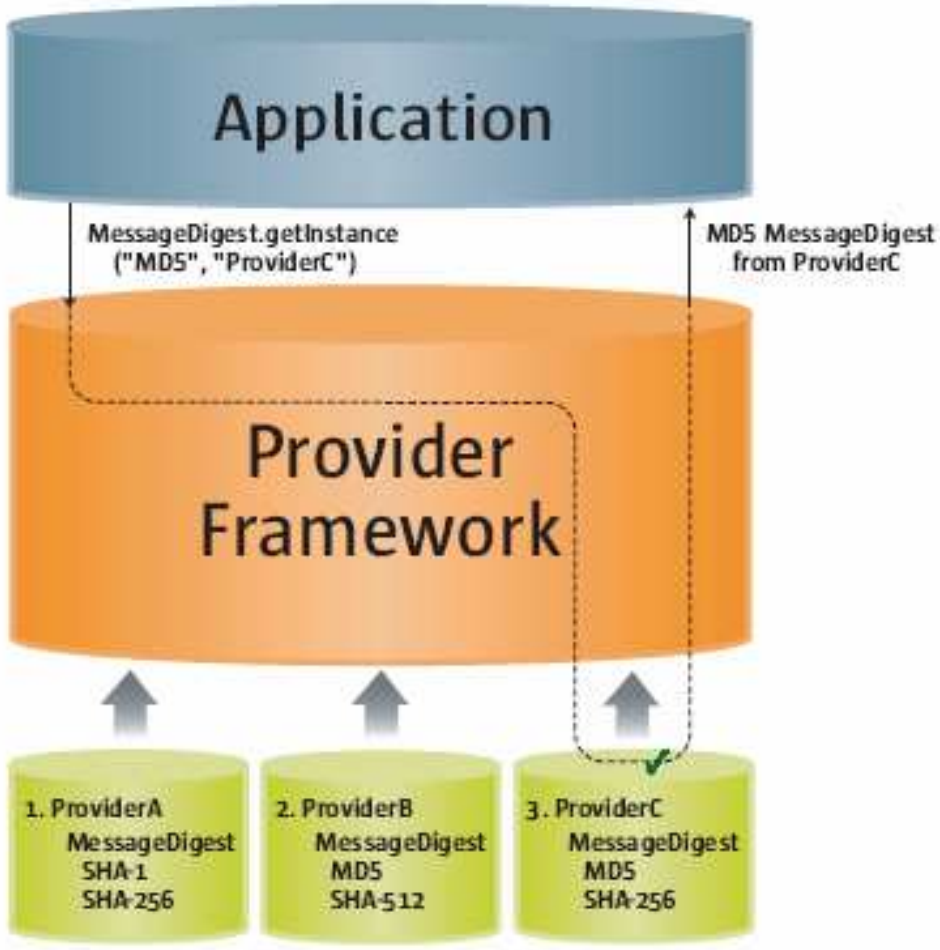


Figure 2 – Specific provider requested

Provider-based Architecture

- Scelta consapevole del provider
  - il nome del provider viene specificato come parametro all'interno del metodo `getInstance()` utilizzato per istanziare l'oggetto richiesto
- Esempi
  - `MessageDigest hash = MessageDigest.getInstance("SHA-1", "BC");`
  - `KeyGenerator kGen = KeyGenerator.getInstance("TripleDES", "BC");`

Installazione di un CSP

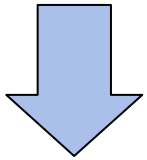
- Installazione statica
  - inserire l'archivio jar che implementa il provider tra le librerie esterne del JRE
    - <*JAVA_HOME*>*jre**lib**ext*
  - modificare il file **java.security** che si trova nella directory <*JAVA_HOME*>*jre**lib**security* aggiungendo il nome del provider come ultimo elemento della lista che ha il formato
    - security.provider.*n=provider*
- Esempio
  - `security.provider.7 = org.bouncycastle.jce.provider.BouncyCastleProvider`

Installazione di un CSP

- Installazione dinamica
  - il provider va importato all'interno dell'applicazione
  - il metodo `addProvider(String nome)` della classe `java.security.Security` consente di aggiungere il provider desiderato a run-time
- Esempio
  - `import java.security.Security;`
  - `import org.bouncycastle.jce.provider.BouncyCastleProvider;`
  - `Security.addProvider(new BouncyCastleProvider());`

Provider-based Architecture

- Nessuna preferenza nella scelta del provider
  - il nome del provider non viene specificato come parametro all'interno del metodo `getInstance()` utilizzato per istanziare l'oggetto richiesto



- Ricerca ordinata per preferenze
  - file *java.security*
  - la ricerca continua finché non si trova il primo provider che implementa la funzionalità richiesta

Provider Sun

- Integrato nel JDK
- Include implementazioni di:
  - algoritmo DSA
    - KeyPairGenerator per DSA
    - AlgorithmParameter per DSA
    - AlgorithmParameterGenerator per DSA
    - KeyFactory per DSA
  - algoritmi MD-5 e SHA-1 (hash)
  - algoritmo SHA1PRNG (SecureRandom)
  - CertificateFactory per X.509 e CRLs
  - KeyStore proprietario JKS

Provider BouncyCastle

- Non integrato nel JDK (va installato)
- Implementa ed estende le tecniche crittografiche definite dalla JCA e dalla JCE
  - implementazione alternativa degli algoritmi di cifratura (AES, DES, TripleDES, PBE, ...)
  - algoritmi di cifratura asimmetrica (RSA, ElGamal)
  - algoritmi di firma digitale
  - padding (None, PKCS1PADDING)
  - facilities per diversi protocolli di KeyAgreement
  - facilities per utilizzare cifrari simmetrici a flusso e a blocchi
  - ...

Esercitazione n.1: PRNG e funzioni hash

Sicurezza dell'Informazione M

A.A. 2011 - 2012

Anna Riccioni

anna.riccioni@unibo.it


Tecniche di
codifica dei dati

Encoding

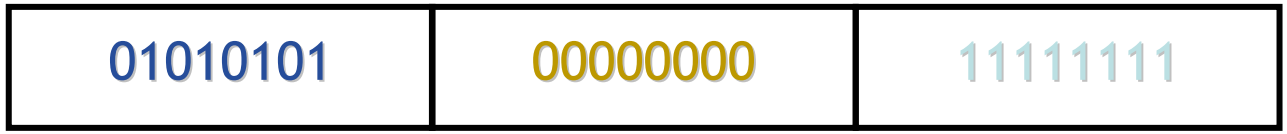
- Codifiche disponibili:
  - UTF-8
  - UTF-16

  stringhe, file di testo
  - HEX
  - BASE-64

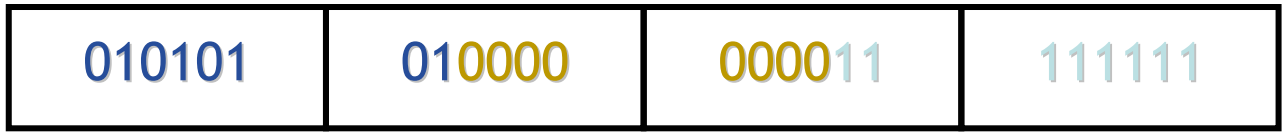
  file binari, file in formati complessi
- HEX e BASE-64 consentono di rappresentare in formato testuale il contenuto di byte che le codifiche ASCII o UTF associano a caratteri non stampabili

BASE-64

- Codifica in BASE-64:
  - anziché considerare un byte alla volta, se ne considerano 3



  - i 24 bit che compongono i 3 byte possono essere suddivisi in 4 blocchi da 6 bit l'uno

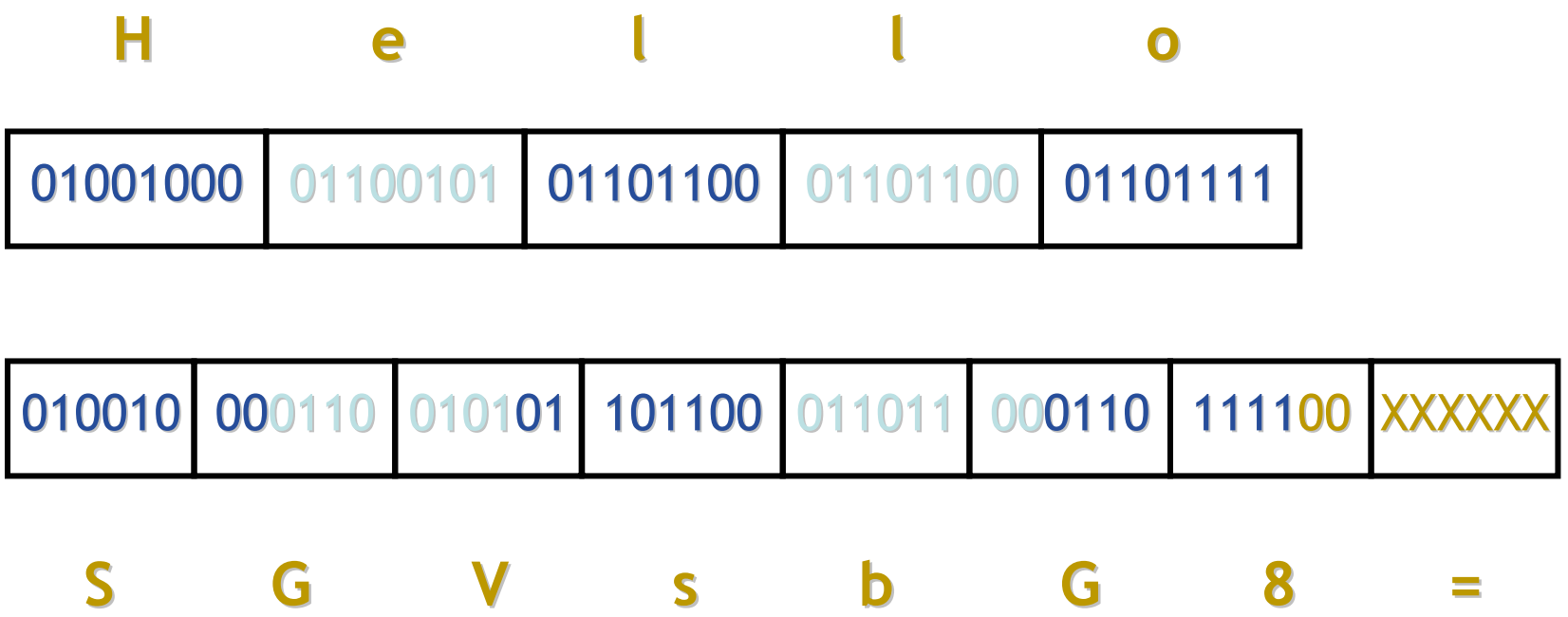


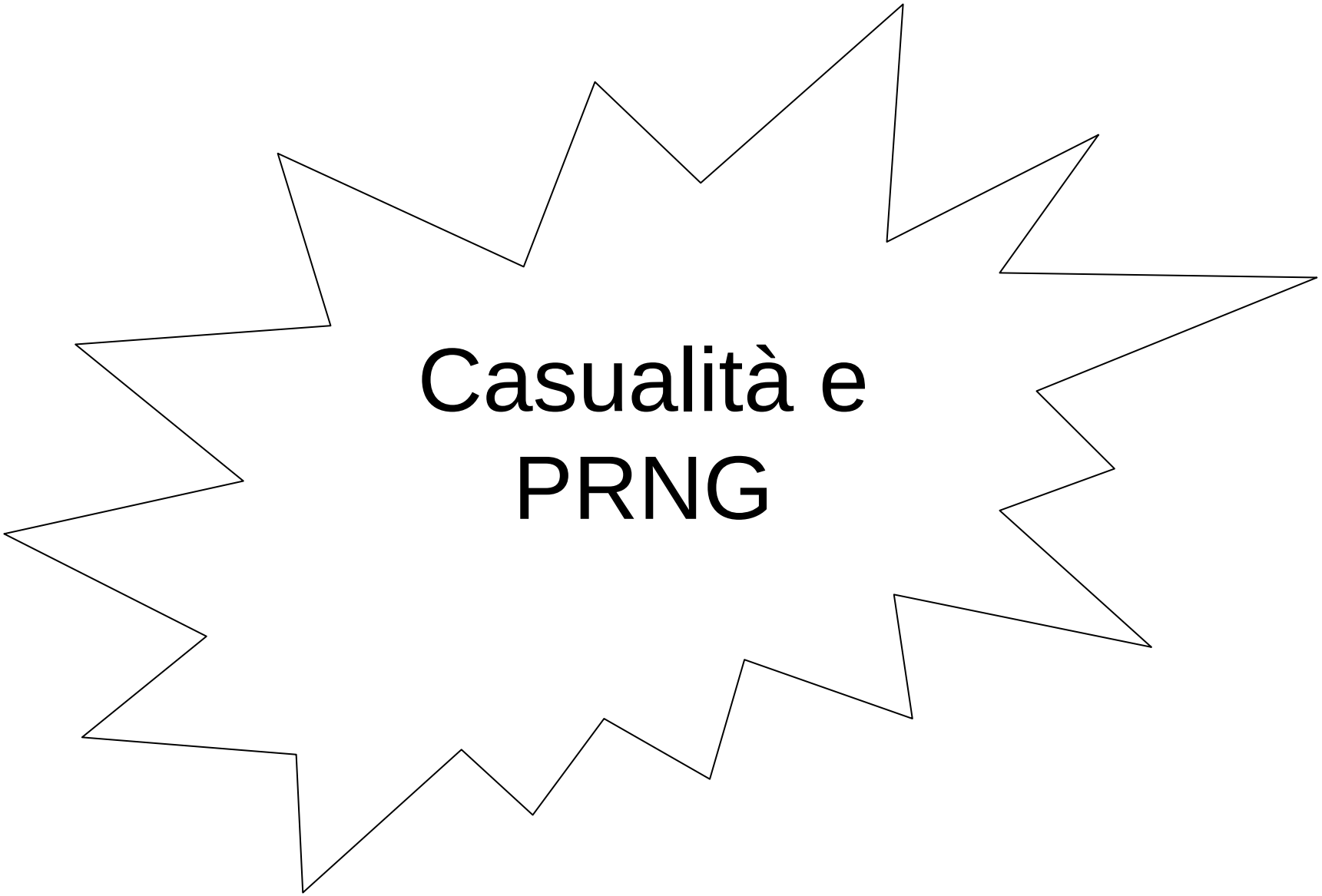
  - un blocco di 6 bit ha 2^6 possibili configurazioni, ognuna delle quali è associata, nella codifica BASE-64, ad uno dei 64 caratteri stampabili nell'insieme [A..Z] U [a..z] U [0..9] U {+} U {/}

BASE-64

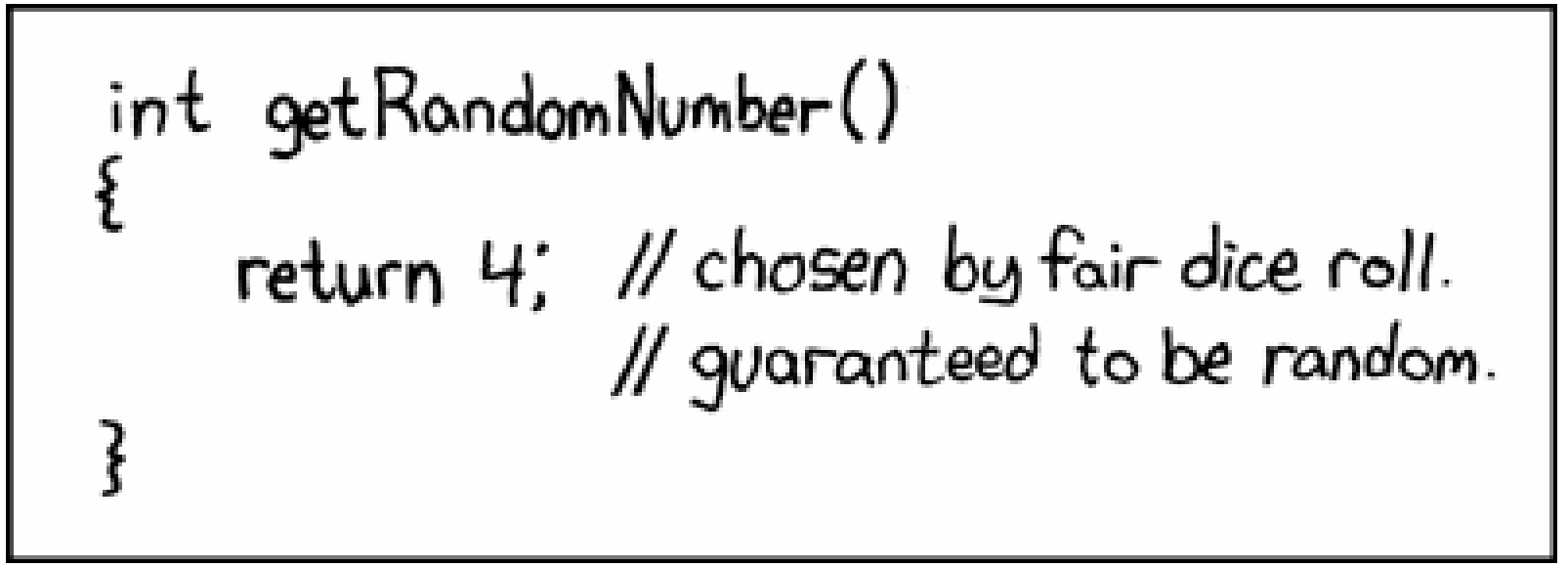
- una codifica in BASE-64 trasforma la sequenza di byte di partenza in una nuova sequenza di byte, leggibile in forma testuale, ma la cui dimensione risulta incrementata di un terzo
- poiché si considerano 3 byte per volta, nel caso in cui la sequenza di byte di partenza non abbia una lunghezza multipla di 3 occorre utilizzare un "**padding**":
  - un eventuale blocco in BASE-64 incompleto viene riempito di "0" fino a raggiungere la dimensione di 6 bit
  - eventuali blocchi da 6 bit mancanti vengono sostituiti dal carattere speciale "="

BASE-64: esempio




Casualità e
PRNG

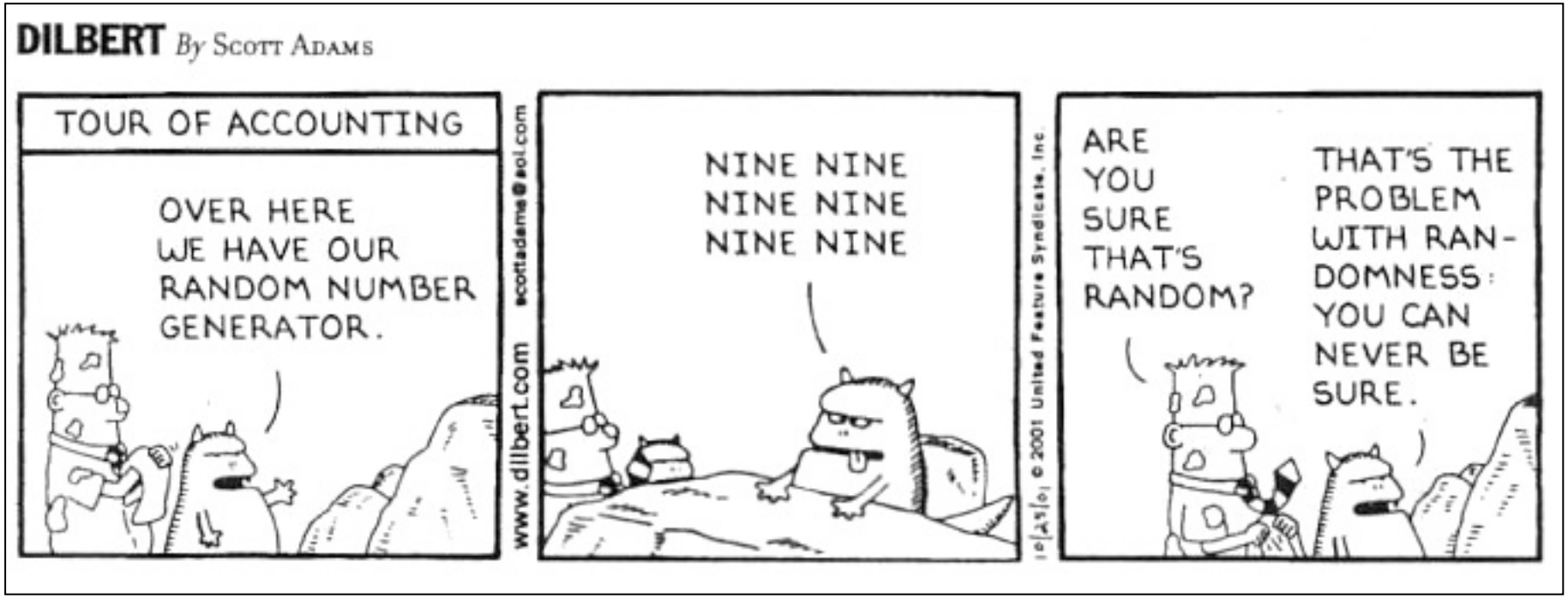
Generazione di numeri (pseudo)casuali



- SecureRandom (Java)
  - RNG / PRNG

Qualità del RNG / PRNG

- Verifica della casualità di una sequenza di bit prodotta da un generatore di numeri casuali



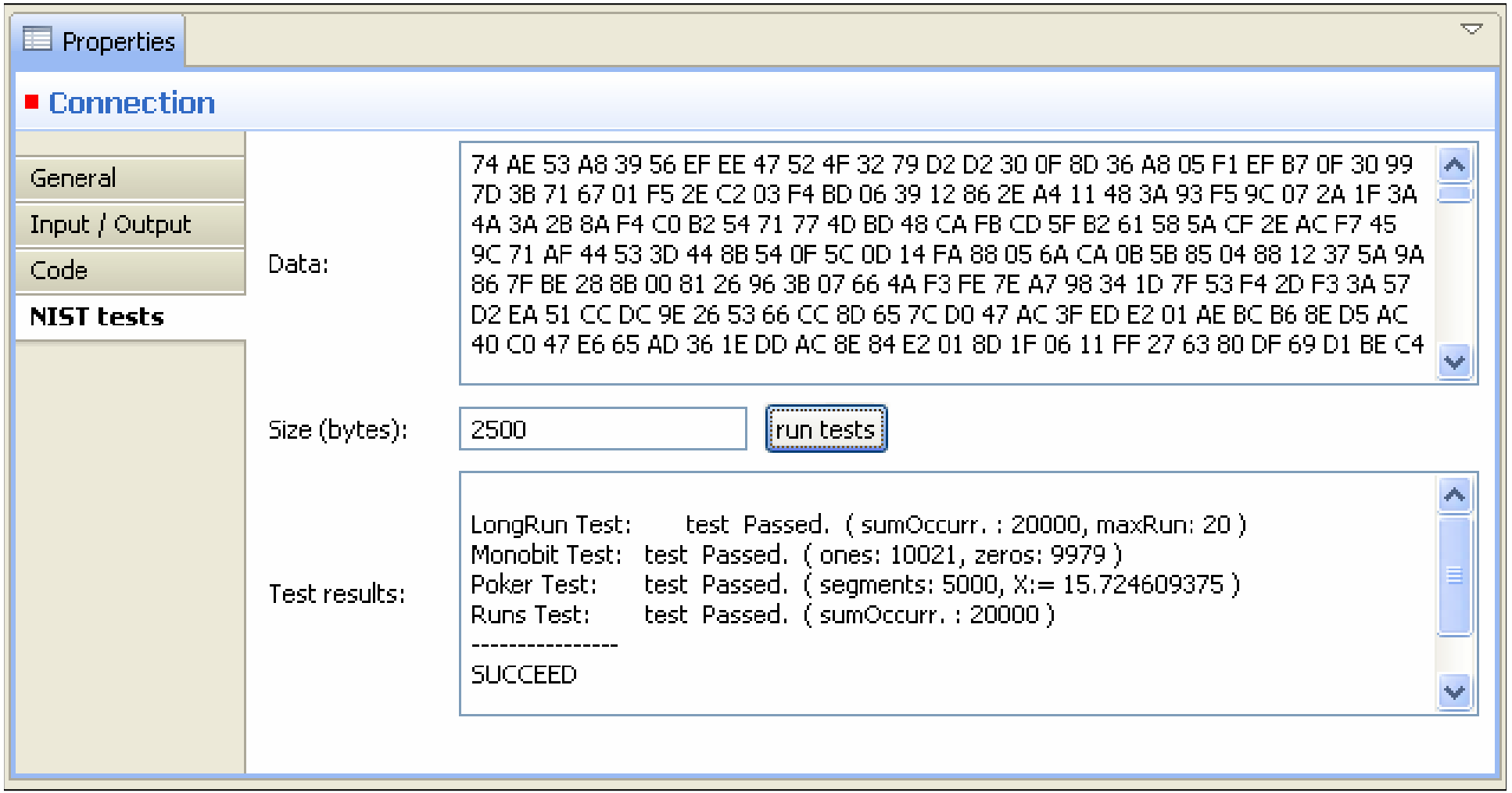
Test statistici

- Verifica della casualità di una sequenza di bit prodotta da un generatore di numeri casuali
- Standard FIPS 140-1 (NIST 1994):
  - campione di 20000 bit
  - superamento di quattro test
    - **monobit**: occorrenze di “0” e “1” circa equivalenti
    - **long run**: corsa più lunga di dimensione inferiore a 26
    - **runs**: occorrenze di tutte le corse di varie dimensioni (1, 2, 3, 4, 5, 6 e superiori) incluso in intervalli prefissati che rispecchiano i valori che si avrebbero in una sequenza casuale
    - **poker**: test del chi quadro per valutare se le occorrenze delle possibili stringhe di 4 bit approssimano il valore atteso per una sequenza casuale

Test statistici

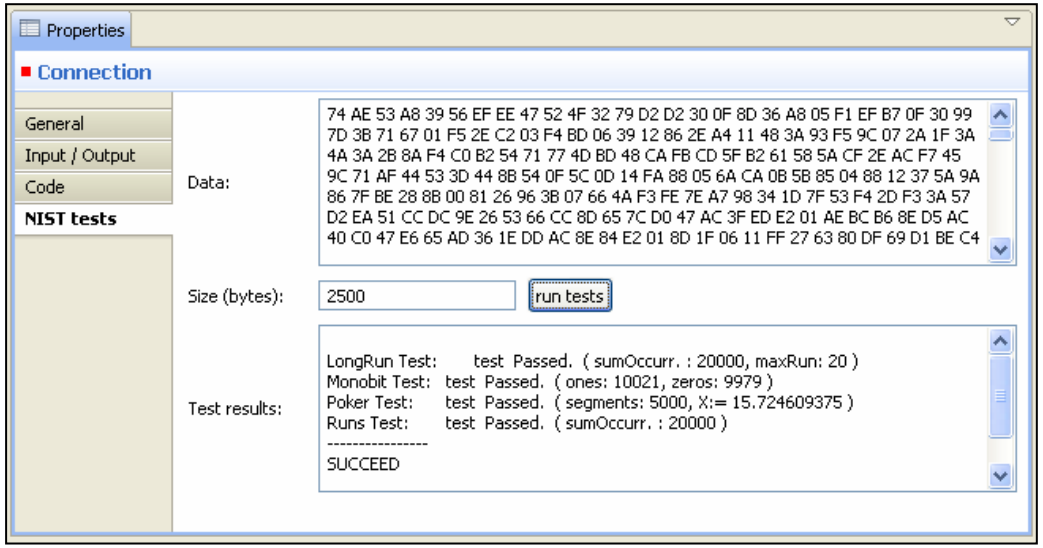
- Prerequisiti
  - standard FIPS 140-1: campione di 20000 bit
  - in S-vLab:
    - i test del NIST possono essere attivati dalla scheda “NIST tests” presente nella vista delle proprietà quando si seleziona una connessione
    - la connessione deve contenere un dato di almeno 2500 byte (=20000 bit)
- Quando utilizzarli
  - verifica dell’output di un generatore di numeri pseudocasuali
  - analisi dei byte prodotti da un’operazione di cifratura
  - …

Test statistici



Test statistici

- Come interpretare i risultati
  - confronto dei parametri caratteristici dei test con quelli considerati ammissibili dal FIPS 140-1

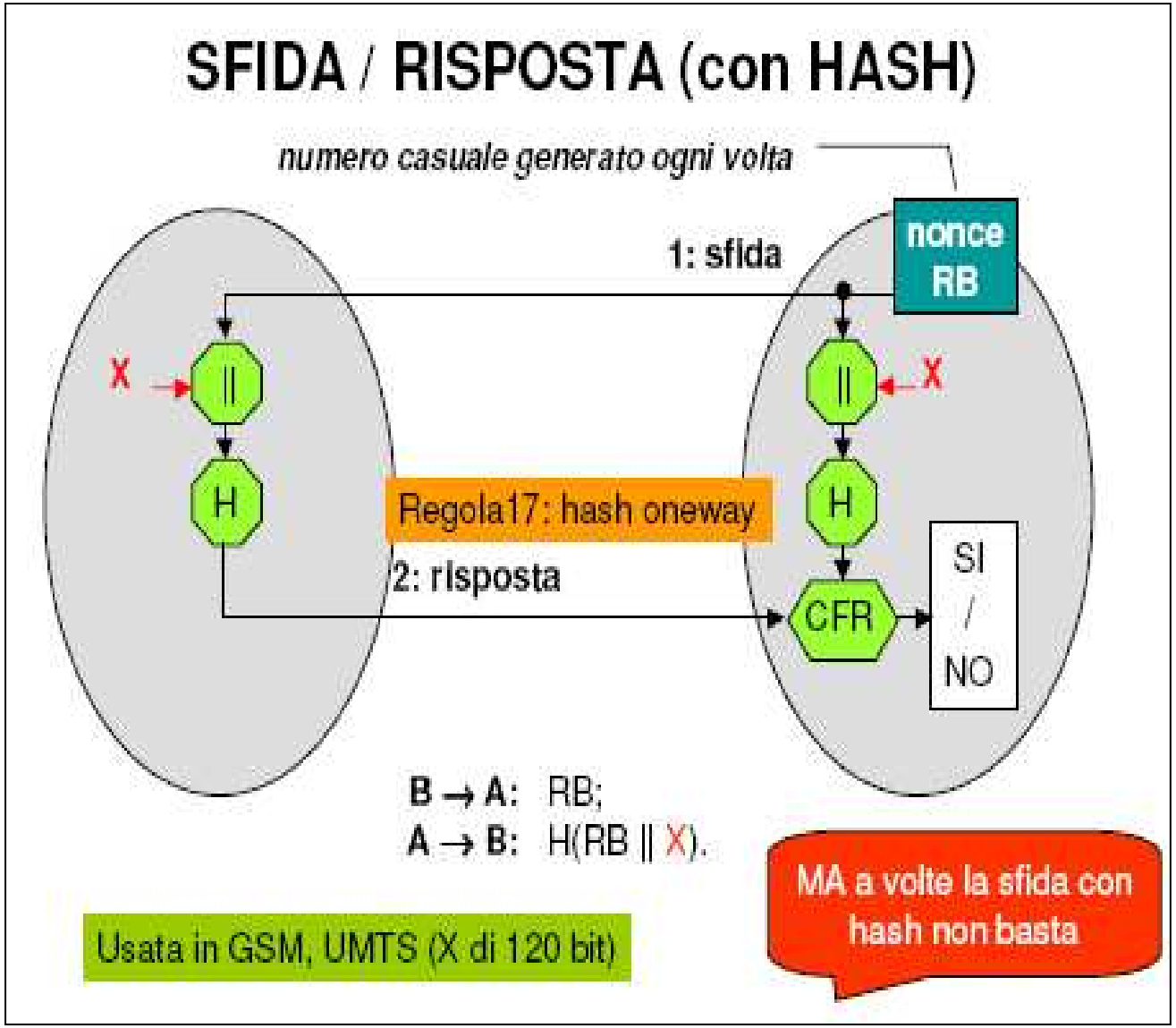


- **long run**: X < 26
- **monobit**: 9725 < X < 10275
- **poker**: 2.16 < X < 46.17
- **runs**:

Lunghezza della corsa	Intervallo ammissibile
1	2315 - 2685
2	1114 - 1386
3	527 - 723
4	240 - 384
5	103 - 209
6+	103 - 209


Altri
esperimenti
possibili

Autenticazione



HMAC (RFC 2104)