MTAT.07.017
Applied Cryptography

Rakenduslik krüptograafia
Прикладная криптография

Juri Hudolejev
University of Tartu
Spring 2011

Topics for This Week

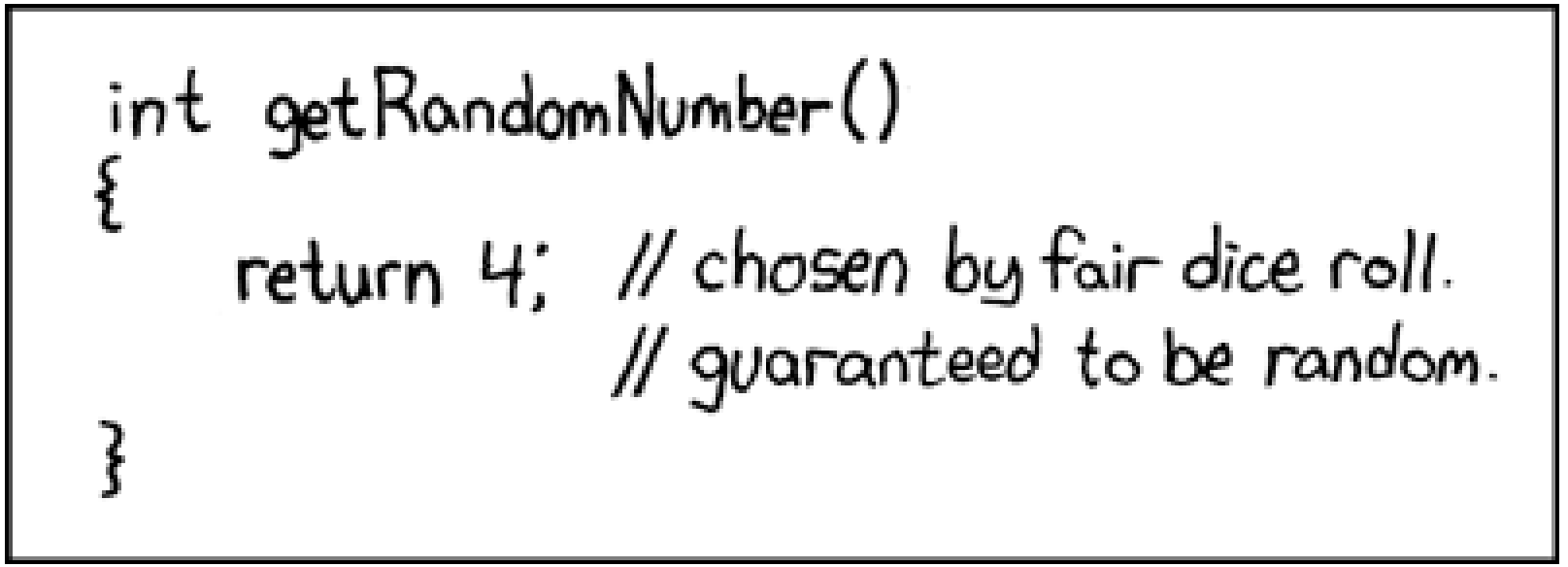
Randomness

Hash functions

Java Cryptography Architecture

Symmetric encryption

Random Numbers?



http://xkcd.com/221/

Random Sequence

Can be described as:

- Sequence of numbers that do not follow any deterministic pattern
- None of the numbers can be predicted based on previous numbers
- Has no description shorter than itself
- Sequence of bits that cannot be compressed

No universal definition (that I am aware of)

Random Sequence

Good one: not (easily) predictable

Are the following sequences predictable?
What is the next element in each case?

4 1 5 9 2 6 5 3

P N U S J M E V

1 1 0 1 0 0 1 1 0 1 0 0 1 0 0 1

Random Generator

No computer program nor system is capable to produce a true randomness

Instead, pseudorandom generators are used

- PRNG – PseudoRandom Number Generator
- DRBG – Deterministic Random Bit Generator

PRNG produces sequence of numbers that has a set of properties similar to those of truly random sequence

Pseudorandom Generators

Linear congruential generators

- Simple arithmetic operations used

Stream ciphers

Block ciphers

Collecting entropy from the environment

- Best pseudorandom sequences

Further Reading

http://en.wikipedia.org/wiki/Randomness

http://www.random.org/randomness/

Also check http://www.random.org/ for more links

http://stackoverflow.com/questions/3956478/

Nice question and answers explaining which of `rand()` and `rand()*rand()` is more 'random'.

Questions?

Exercise 2.1

Lottery :)

Generate 6 random integers: R_1 to R_6
so that $1 \leq R_i \leq 36$

Modify your code to produce exactly the same numbers on every run

Hash Function

Procedure to convert data of variable length to a fixed-length bit string

Related (and confusing) terms:

- **Hash table** (hash map)
  - Structure that maps search keys to elements (phone book?)
- **Fingerprint**
  - Used to detect data modification
- **Checksum**

  Used to detect transport errors

Cryptographic Hash Function

Message == initial (raw) data

(Message) Digest == hash value

- Computing message digest is easy (and fast)
- Difficult to restore message from digest
- Difficult to find two different messages with the same digest
- Even small message modification results a big difference in digest values

Message Digest: Example

Message:

Code runs faster when Chuck Norris watches it

MD5:

`a7907265ebbcd38501e1ba75161eed72`

SHA-1:

`3d7a9d2c920594b1fa91fa59a7c548d70577524a`

RIPEMD-160:

`c4b579a68c85a5a547b2c9c87921659aefbf42a0`

Message Digest: Java Example

```
String message = "Code runs faster " +
   "when Chuck Norris watches it";

MessageDigest messageDigest =
   MessageDigest.getInstance("SHA");

byte[] digest =
   messageDigest.digest(message.getBytes());
```

Exercise 2.2

Compile and run the code from previous slide

What other hash algorithms can you use?

* * *

Get Bouncy Castle crypto API for Java from
http://www.bouncycastle.org/latest_releases.html
You will need `bcprov-jdk16-145.jar`

Java Cryptography Architecture

Framework for working with cryptography

- Key and certificate management
- Encryption and decryption (also hashing)
- Secure random generators

Introduced in JDK 1.1 in 1997

Part of `java.security` package

Provider-based architecture

Provider-based Architecture

Very abstract description:

- Certain functionality is implemented in special classes, packages, etc. **– Cryptographic Service Provider**
- Providers are registered within system and 'inform' it that they implement functionality X, Y and Z.
- If system wants to use the functionality X, it choses one of the providers and uses implementation it contains.

Provider Example

```
// SHA-256 implementation
public class MyDigest extends MessageDigest { … }

// Provider description
public class MyDigestProvider {
  public MyDigestProvider() {
    put("MessageDigest.SHA-256", "MyDigest");
  }
}

// Using the provider
Security.addProvider(new MyDigestProvider());
MessageDigest messageDigest =
    MessageDigest.getInstance("SHA-256");
```

Task 2.1 (2 points)

Implement a simple command-line utility to compute and verify hash values of files

- Algorithms: MD5, RIPEMD-160, SHA-1, SHA-256

Usage:

- `java Hasher <ALG> <FILE>`
- Output: `ALG` and hash value in hex (base 16)
- `java Hasher <ALG> <FILE> <DIGEST>`
- Output: 'hash values match / do not match'

Task 2.2 (4 points)

Implement simple command-line utilities to encrypt and decrypt files using symmetric encryption schemes

Keywords: AES, PBE

Usage:

```
java Encrypt <FILE> <ENC> <PASS>
java Decrypt <ENC> <FILE> <PASS>
```

Questions?

Happy Coding!

Tasks, exercises, additional info:

http://courses.cs.ut.ee/2011/appcrypto/Main/Lab02

Deadline: 2011-02-25 08:00 EET

Contact: Juri Hudolejev <juri@ut.ee>

Next lab session:

Friday 2011-02-25 **08:30** EET @ Liivi 2 - 205