PA193 - Secure coding principles and practices

Team project: bip39 mnemonic phrase generator (and verifier)

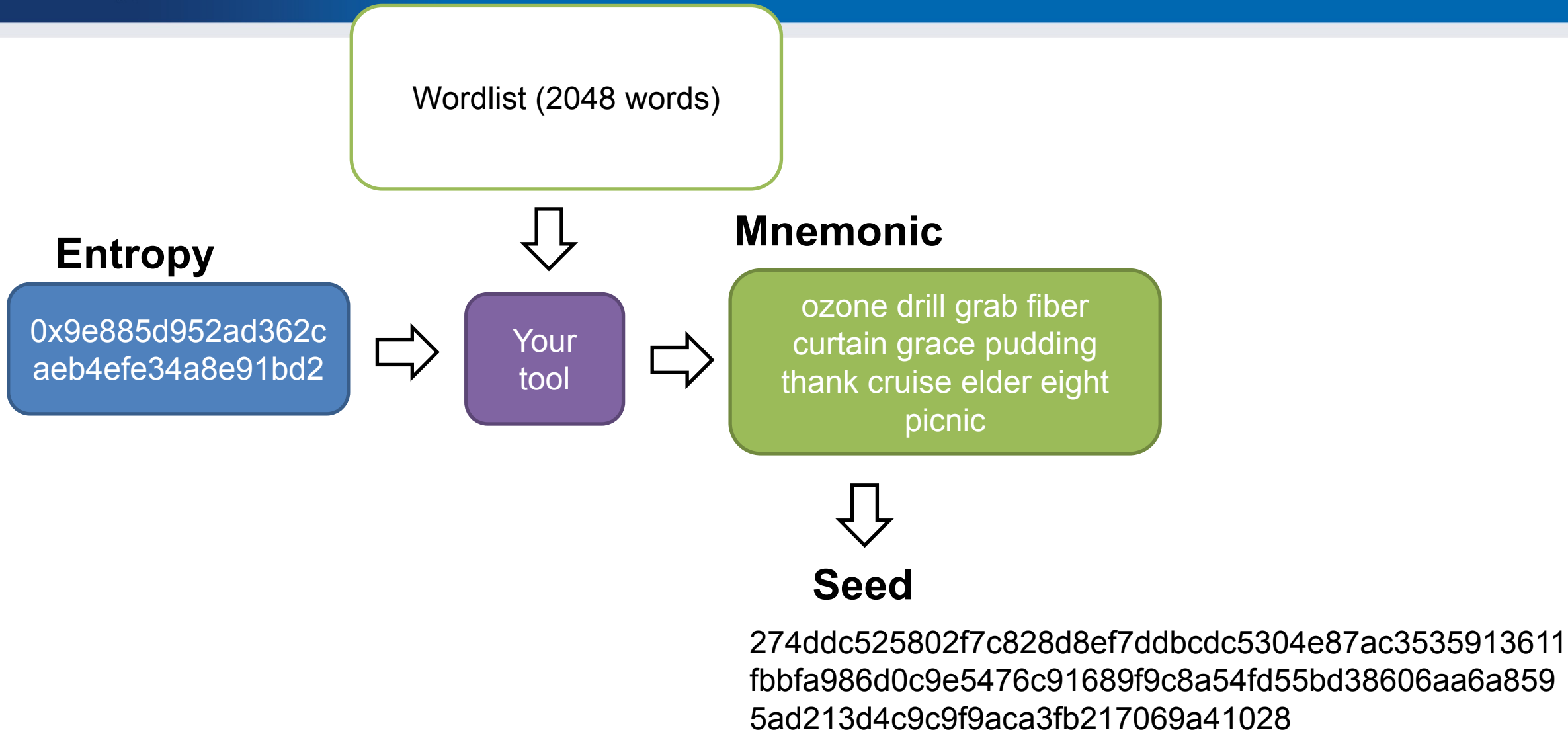
Petr Švenda *svenda@fi.muni.cz* *@rngsec*
Centre for Research on Cryptography and Security, Masaryk University

CROCS
Centre for Research on Cryptography and Security

Project idea

1. Write (securely) generator and verifier of bip39 mnemonic phrases
2. Write code as library code with proper API + demonstration usage
3. Analyze own implementation with static and dynamic analysis tools
4. Use GitHub + TravisCI integration for automatic tests, test vectors
5. Review code of other team generator
6. Create patch for selected flaws and open pull request

- https://github.com/bitcoin/bips/blob/master/bip-0039.mediawiki


Wordlist (2048 words)
Entropy
0x9e885d952ad362caeb4efe34a8e91bd2
Your tool
Mnemonic
ozone drill grab fiber curtain grace pudding thank cruise elder eight picnic
Seed
274ddc525802f7c828d8ef7ddbcdc5304e87ac3535913611fbbfa986d0c9e5476c91689f9c8a54fd55bd38606aa6a8595ad213d4c9c9f9aca3fb217069a41028

Required functionality: entropy → seed and phrase

- Input:
  - entropy (binary or hexadecimal string), “9e885d952ad362caeb4efe34a8e91bd2”
- Output:
  - text mnemonic phrase: “ozone drill grab fiber curtain grace pudding thank cruise elder eight picnic”
  - seed: “274ddc525802f7c828d8ef7ddbcdc5304e87ac3535913611fbbfa986d0c9e5476c91689f9c8a54fd55bd38606aa6a8595ad213d4c9c9f9aca3fb217069a41028”

Required functionality: phrase → entropy and seed

- Input: mnemonic phrase
- Output: seed and initial entropy as hexadecimal strings

Required functionality: phrase & seed → OK/NOK

- Input: mnemonic phrase and expected seed
- Output: OK if provided phrase generates expected seed, NOK otherwise

Required functionality: library code with API

- Create core functionality as library code
  - => easy to use in other projects
- Create program “main()” providing command line interface (CLI)
  - cmd-line arguments, std input, read from files…
- Create demonstration of required functionality
- Document usage properly (README.md)

Required functionality

- Implement required in selected language (must be agreed with PetrS)
  - Any language, but:
    - 2 different tools for static analysis must be available
    - 2 different tools for dynamic analysis
    - No repetition of language within the same seminar group
- You must write code yourself, no use of external code or libraries is allowed
  - Exception: SHA2-512, SHA2-256
- Use defensive programming: only valid inputs are accepted, invalid rejected
- Implement tests, cover also following test vectors
  - https://github.com/trezor/python-mnemonic/blob/master/vectors.json

Teams

- 3 people per team
  - Assigned today (within group)
- Teams must use GitHub for cooperation
  - Distribute work load between all members
  - Contribution from all team members must be visible in commits (git commits from the member)
  - Your evaluation will be partially based on your participation
- Start working early, especially with implementation

Immediate next steps (seminar 3.10.2019)

- Form groups, exchange emails, get together, select your language, find required static and dynamic analysis tooling, book with me now
- Setup repo at GitHub '*PA193_mnemonic_XXX*' where XXX is name of your team
  - Create README.md
- Setup GitHub and TrevisCI integration (first test is code compilation)
- Start writing mnemonic generator implementation (use small commits)

Projects - timeline

1. Write code (GitHub): max. **10 points** (**31.10.2019**)
   - Complete implementation + presentation [31.10.2019, your seminar group]

2. Review, attack, present and patch implementations: max. **10 points** (**16.12.2019**)
   - Initial kickoff together with implementation team [31.10.2019]
   - Write patch for a selected bug, open pull request
   - Report + presentations [16.12.2019, Monday 16:00 – all teams together]

- At least 10 points (total) from project are required

TEAMS

- Team BugsBunny (language C):
  - https://github.com/xx/PA193_mnemonic_xxx
  - Member1 <456271@mail.muni.cz>
  - Member2 <456443@mail.muni.cz>
  - Member3 <394097@mail.muni.cz>
- Team SIGSEGVBoys (language C++):
  - https://github.com/xx/PA193_mnemonic_SIGSEGVBoys
  - Adam Považanec <469045@mail.muni.cz>
  - Tomáš Kancko <469029@mail.muni.cz>
  - Radoslav Sabol 469331@mail.muni.cz
- Team Hypershell (language Python):
  - https://github.com/xx/PA193_mnemonic_xxx
  - Ondřej Krčma
  - Mohannad Yousef
  - Yorick Kupczyk
- Team BigBugs (language Java):
  - https://gitlab.fi.muni.cz/xdlhopol/PA193_mnemonic_bigbugs
  - Martin Horáček
  - Daniel Dlhopolček
- Team Aller (language Go):
  - https://gitlab.fi.muni.cz/xdlhopol/PA193_mnemonic_bigbugs
  - Olivier Bal-Petre 497391
  - Jan Kvapil 408788
  - Alexandre Le Clanche 497577

- Team Bernard's Star (language C++):
  - https://github.com/xx/PA193_mnemonic_xxx
  - Maximilian Kindt <497552@mail.muni.cz>
  - Isaac Padberg <496498@mail.muni.cz>
  - Martin Vondráček <17825@mail.muni.cz>
- Team xxx (language C):
  - https://github.com/xx/PA193_mnemonic_xxx
  - Evann De bailliencourt<497457@mail.muni.cz>
  - Stephane Hamaili <497728@mail.muni.cz>
  - Member3 <394097@mail.muni.cz>
- Team CAPSLOCKGANG (language C#):
  - https://github.com/xx/PA193_mnemonic_xxx
  - Martin Gregorik<500359@mail.muni.cz>
  - Henrich Horvath <409990@mail.muni.cz>
  - Member3 <394097@mail.muni.cz>
- Team lazy beancounters (language Rust):
  - https://github.com/dufkan/PA193_mnemonic_trusty_rusty
  - Jakub Bartolomej Kosuth<433295@mail.muni.cz>
  - Daniel Filakovsky<445605@mail.muni.cz>
  - Antonin Dufka <445281@mail.muni.cz>

- Cyber Warriors (language C):
  - https://github.com/xx/PA193_mnemonic_xxx
  - Ankur
  - Nomit
  - Amal
- Team Slytherin (language Python):
  - https://github.com/xx/PA193_mnemonic_xxx
  - Martin Vondracek
  - Solodkova Elena
  - Samuel Obuch