Institute of Computer Science, University of Bern
Privacy and Data Security, Spring 2024

Prof. Christian Cachin
Mariarosaria Barbaraci

Exercise 10

10.1 Irreversible tokenization (10pt)

An *International Bank Account Number (IBAN)* contains a two-digit checksum to protect it from errors made by humans when copying the number.

An IBAN is a string of up to 34 alphanumeric characters (digits and capital letters). The string contains, in this order, (1) a country code of two letters, (2) two decimal check digits, and (3) an account number of up to 30 characters. Format and length of the account number vary by country. An IBAN in Switzerland consists of 21 characters according to the pattern:

```
CHzz bbbb bccc cccc cccc c
```

Here `CH` denotes the country code, `zz` are the check digits, `bbbbb` denote the bank (Swiss Bank Clearing Number), and `cc...` contains the account number at the bank.

To check the validity of an IBAN string s, perform these four steps:

1. Move the first four characters to the end of s.
2. Replace every letter in s by two digits, according to the rule $A = 10$, $B = 11$, ..., $Z = 36$. Note that s may get longer.
3. Interpret s as a number n in decimal notation.
4. If $n \equiv 1 \pmod{97}$, then s contains a valid IBAN, otherwise not.

Tasks:

a) Implement a function `check_iban(s)` in Python that returns `True` whenever the IBAN in string `s` is valid.

b) Implement a hash-based *irreversible tokenization* function `hash_token(s)` that takes valid Swiss IBAN in `s` and returns another valid Swiss IBAN. Use the cycle-walking method and SHA-256 from Python's `hashlib`.

c) Extend your tokenization function to a *keyed* irreversible tokenization function `mac_token(key, s)`, which additionally takes an arbitrary string `key` as input that serves as the key. Use HMAC-SHA256 from Python's `hmac` library and the cycle-walking method.

This description suggests to use Python, but you may also use Java or C++ as programming language. For further languages, check with the assistants.

SHA-256 returns an array of 32 bytes, which is not in the correct format for cycle walking: interpret this as a number in binary, reduce it modulo 10^n, and convert the truncated number to an n-digit string. Python's `hashlib.sha256.hexdigest()` gives a 32-byte string `s` that one can convert to an integer using `int(s, base=16)`, for instance.

10.2 Reversible tokenization with FPE (bonus +5pt)

Format-preserving encryption (FPE) can be used for *reversible* tokenization. The auxiliary file `ex10-smallcipher.py` contains a Python implementation of a small-domain encryption algorithm `smallcipher_encrypt`, according to Black and Rogaway [BR02] and Bellare *et al.* [BRRS09, Scheme FE2/FD2, Fig. 3]

a) Implement an FPE-based *reversible tokenization* function `fpe_encrypt(key,tweak, s)` to compute an FPE of a string `s` that contains a valid Swiss IBAN. Let `key` be the encryption key and `tweak` be an arbitrary string that "tweaks" the small-domain cipher; it can be thought of a domain separator.

b) Implement the corresponding decryption algorithm and verify that it inverts the encryption.

Use again the cycle-walking method, as developed for Problem 10.1.

References

[BR02] John Black and Phillip Rogaway. Ciphers with arbitrary finite domains. In Bart Preneel, editor, *Topics in Cryptology - CT-RSA 2002, The Cryptographer's Track at the RSA Conference, 2002, San Jose, CA, USA, February 18-22, 2002, Proceedings*, volume 2271 of *Lecture Notes in Computer Science*, pages 114–130. Springer, 2002. `doi:10.1007/3-540-45760-7\_9`.

[BRRS09] Mihir Bellare, Thomas Ristenpart, Phillip Rogaway, and Till Stegers. Format-preserving encryption. In Michael J. Jacobson Jr., Vincent Rijmen, and Reihaneh Safavi-Naini, editors, *Selected Areas in Cryptography, 16th Annual International Workshop, SAC 2009, Calgary, Alberta, Canada, August 13-14, 2009, Revised Selected Papers*, volume 5867 of *Lecture Notes in Computer Science*, pages 295–312. Springer, 2009. `doi:10.1007/978-3-642-05445-7\_19`.