Implementing Cryptographic Primitives for BlockChain

Cryptography CS 411& CS 507 Term Project for Fall 2018

E. Savaş
Computer Science & Engineering
Sabancı University
İstanbul

Abstract

You are required to develop essential building blocks of cryptocurrency using block chains.

1 Introduction

The project has three phases:

- Developing software for digital signature
- Developing software for proof of work
- Developing software for other building blocks and integration

More information about the phases are given in the subsequent sections.

2 Phase I: Developing software for digital signature

In this phase of the project, you will develop software for signing given any message. For digital signature (DS) you will us an algorithm, which consists of four functions as follows:

- **Public parameter generation:** Two prime numbers p and q are generated with $q|p-1$, where q and p are 224-bit and 2048-bit integers, respectively. The generator g generates a subgroup of $\mathbb{Z}_p^*$ with q elements. Naturally, $g^q \equiv 1 \bmod p$. Note that in your system q, p, and g are *public parameters* shared by all users, who have different secret/public key pairs. Refer to the slide (with title "DSA Setup" in chapter 10 for an efficient method for parameter generation).
- **Key generation:** A user picks a random secret key $0 < \alpha < q$ and computes the public key $\beta = g^\alpha \bmod p$.
- **Signature generation:** Let m be an arbitrary length message. The signature is computed as follows:
    1. $k \leftarrow \mathbb{Z}_q$, (i.e., k is a random integer in $[0, q-1]$).

2. $r = g^k \pmod{p}$
3. $h = \text{SHA3\_256}(m||r)$
4. $s = \alpha \cdot h + k \pmod{q}$
5. The signature for m is the tuple (s, h).

- **Signature verification:** Let m be a message and the tuple (s, h) is a signature for m. The verificiation proceeds as follows:
  - $v = g^s \beta^{-h} \pmod{p}$
  - $\tilde{h} = \text{SHA3\_256}(m||v)$
  - Accept the signature only if $h = \tilde{h} \bmod q$
  - Reject it otherwise.

Note that the signature generation and verification of this DS are different than those discussed in the lecture.

You are required to develop Python software that implements those four functions; namely SETUP for public parameter generation, KEY GENERATION, SIGNATURE GENERATION and SIGNATURE VERIFICATION. You are required to test your software using the test routines in "DS_Test.py" provided in the assignment package.

In "DS_Test.py", there are four basic test functions:

1. CHECKDSPARAMS(q, p, g) takes your public parameters (q, p, g) and check if they are correct. It returns 0 if they are. Otherwise, it returns a code that indicates the problem.

2. CHECKKEYS(q, p, g, α, β) takes your public parameters (q, p, g) and key pair (α, β) and check if the key pair is correct. It returns 0 if they are; otherwise it returns -1.

3. CHECKSIGNATURE(q, p, g, α, β) takes your public parameters (q, p, g), key pair (α, β) and generates a signature for a random message and verifies the signature. It returns 0 if the signatures verifies; otherwise it returns -1.

4. CHECKTESTSIGNATURE() reads the file "TestSet.txt" (provided in the assignment package), which contains public parameters, a public key, and 10 randomly chosen messages and their signatures. The test code reads them and runs signature verification function. The test code returns 0 if all signatures verify; otherwise it returns -1.

In this phase of the project, you are required to upload only a file named "DS.py" with sufficient comments. We will be able to test your code using "DS_Test.py'. If your software cannot be tested by "DS_Test.py' as it is, you will get no credit.

3 Appendix I: Timeline & Deliverables & Weight & Policies etc.

Project Phases	**Deliverables**	**Due Date**	**Weight**
Project announcement		07/12/2018	
First Phase	Source code (DS.py)	14/12/2018	25%
Second Phase	TBA	21/12/2018	TBA
Third Phase	TBA	28/12/2018	TBA
Bonus	TBA	28/12/2018	TBA

3.1 Policies

- You may work in groups of two.
- You may be asked to demonstrate a project phase to a TA or the instructor.
- In every phase, we will provide you with a validation software in python language that can be used to check your implementation for correctness. We will also use it to check your implementation. If your implementation in a project phase fails to pass the validation, you will get no credit for that phase.