CS306: Introduction to IT Security (Fall 2017)
Homework #2

Instructor: Nikos Triandopoulos

November 12, 2017

Instructions

Please carefully read the following guidelines on how to complete and submit the homework.

1. This homework is due on **Sunday, November 19, 2017, at 11:59pm**.
2. No submissions will be accepted after **Tuesday, November 21, 2017, at 11:59pm**. Late submissions will contribute (one or two) late days to your five-late-days limit (see announcement on Canvas related to the updated late handin policy).
3. You should handin this assignment via Canvas as a .zip archive that contains your source code for any programming problems and a .pdf document with your typed solutions for all other problems. Name your submitted .zip file using the 'LastName_FirstName' format (e.g., as `Triandopoulos_Nikos.zip`), which should expand to a directory with the same name.
4. You are bound by the Stevens Honor System. There is no collaboration allowed for this homework. You may use any sources related to course materials, but information from external sources must be properly cited.
5. Solving correctly all five problems amounts to 110% of the total homework grade, i.e., there is an opportunity for a 10% extra credit.

Problem 1: Domain-extension MAC implementation (30%)

Given the provided support code in Java, implement a secret-key message authentication code that employs only a block cipher (and no other cryptographic primitive) to authenticate messages of any size in a *bandwidth-efficient manner*. In particular and as specified in the provided instructions:

(1) Implement the `mac()` and `verify()` methods.

(2) Demonstrate that they are correct by providing the MAC tag (in hexidecimal) of the specified default message using the specified default key.

(3) Explain which algorithm you implemented and why.

(4) Explain what are the domain-extension features of your algorithm in relation to its security.

> Hint: Does your implementation securely handle messages of fixed size, messages of any size or messages of any fixed size?

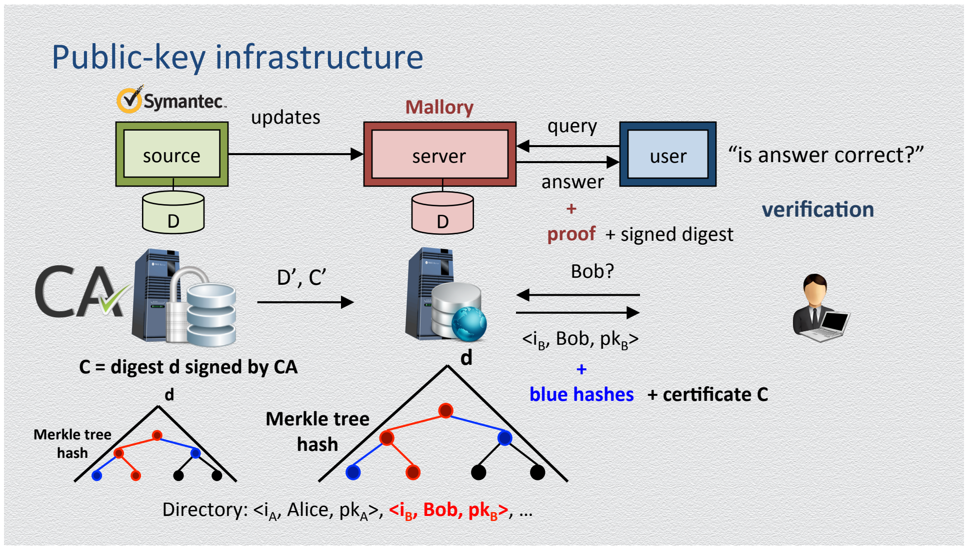


Figure 1: The public-key dictionary-as-a-service model for verifying public keys.

Problem 2: Data outsourcing & public-key infrastructure (20%)

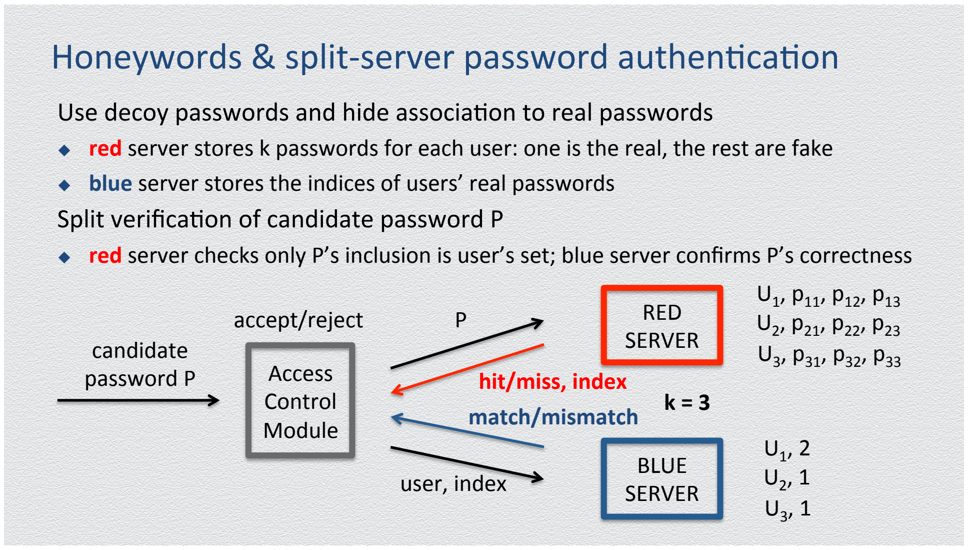
(1) To protect the secrecy of course-related communications, CS306 makes use of public-key encryption: Enrolled students and staff members have their public keys registered with a trusted certification authority (CA), e.g., Symantec; that is, each CS306 person with Stevens UID i and name n_i has a public key pair (sk_i, pk_i). For efficiency reasons, the CA makes the directory $D = \{(i, n_i, pk_i) | i \in \text{CS306}\}$ of all such public keys available (for people to query) through a Stevens online service that is administered by Mallory. Specifically (see also Figure 1):

- The CA provides Mallory with the public-key directory D along with a special certificate C that is the Merkle-tree digest of the directory signed by the CA.
- To send a confidential message to Bob, Alice asks Mallory for his public key—even if Alice had recently learned his public key via a previous query to Mallory, since public-key pairs can be occasionally refreshed or revoked.
- Along with Bob's public-key record (i_B, Bob, pk_B) in D, Mallory also returns to Alice the certificate C and a Merkle-tree proof corresponding to Bob's record.
- After any change in the class enrollment (e.g., a student drops it or enrolls in it with delay) or any key pair is refreshed, the CA provides Mallory with the new (that is, updated) directory D' and the new (that is, corresponding to D') certificate C'.

Suppose that Eve manages to secretly get access to Bob's laptop and successfully steal its secret key sk_B. When Bob becomes suspicious of this, he registers a new public-key pair with the CA.

How can Eve collaborate with Mallory in order to decrypt all subsequent messages sent to Bob? What is the name of this attack type?

(2) Describe how the use of periodically timestamped signatures (i.e., signatures on a timestamped message) can be employed by the CA to provide a solution to the above attack. You can assume that no public key will be updated twice within the same day, and thus consider a 1-day period.



Honeywords & split-server password authentication

Use decoy passwords and hide association to real passwords
- red server stores k passwords for each user: one is the real, the rest are fake
- blue server stores the indices of users' real passwords

Split verification of candidate password P
- red server checks only P's inclusion is user's set; blue server confirms P's correctness

Figure 2: Hardening password security by employing decoy passwords in a split-server architecture.

Problem 3: Intrusion resilience (30%)

Honeywords comprise a recently proposed method for hardening the password security against stolen password files (after an attacker compromises an authentication server). The idea is to distribute password verification across two servers, say one red and one blue, each storing and verifying "half" of the credentials needed to verify in order to successfully authenticate a user (see also Figure 2). Whereas the final accept/reject decision depends on both the red and the blue partial decision, compromising any one server alone provides no (or little) advantage to an attacker.

(1) How does this split architecture improve security? Consider the two cases where an attacker compromises only one of the servers.

(2) Explain how honeywords make password cracking detectable.

Hint: Users are not aware of the existence of honeywords.

(3) What constitutes a good honeyword for a user whose real password is `paSSword5`, if honeywords are generated by tweaking real passwords, i.e., by keeping the main password structure but changing special symbols and numbers?

(4) You just stole the honeywords list of one of the employees in the Office of the Registrar, which consists of passwords: `Blink123`, `Blink128`, `itWb!%s45_3gMoI00286!*mooewTi409##21jUi`, and you have only one chance to impersonate him/her (and try to increase your GPA).

Which password will you choose and why?

Problem 4: On the RSA cryptosystem (30%)

(1) The RSA cryptosystem relies on modular exponentiations, as its core operations. How are such operations realized more efficiently in practice? How is RSA decryption/signing further accelerated?

(2) Given the support code in Python that was provided in Lab#7, implement the RSA key-generation algorithm. Namely, submit the completed skeleton code for Lab#7, including the methods you finished during the lab and the RSA key-generation algorithm, i.e., method `keygen(size)`.