Student unknown
CSCI 1660
Brown University

February 24, 2024

HOMEWORK I — Crypto Party

1. Paper Threat Models

1.1. How is the system insecure?

(I) One of the more obvious adversaries in this system who may want to misuse it is students, who may want to change the rubric to improve their grades. With grades being such a contentious part of the schooling process, very invested students may go to great lengths in an attempt to modify their grades (often due to external and internal pressures put on them by themselves, their parents, and their friends).

(II) One attack that students may want to perform is an attack on the confidentiality of the rubrics. They may try to perform an attacker-in-the-middle attack by getting a look at another student's rubric or report when they are handing in their submission. This may allow them to get an advantage by breaking collaboration policy and seeing what other students are writing in their reports. Additionally, students could perform an attack on the integrity of the submission by modifying the rubric while they are writing up their report and then submitting a falsified final submission. This could allow them to get a higher grade by impersonating the TA's filled-out rubric. Furthermore, a very malicious student could use a similar attack to submit falsified reports for other students in an attempt to sabotage their grades (and maybe achieve a more favorable curve).

1.2. Technique 1: Authentication Measures for Submissions

(I) For this technique, we will have every TA use Message Authentication Codes (MAC) algorithm to combine their MAC key K and the rubric they intend to send in some way that produces a MAC they can send along with the rubric. Only the professor and the TAs will have access to the MAC key K, so they can use it to decode the MAC that comes with any rubric and be sure that the rubric was created and sent by someone else with the key (ie only course staff).

(II) This will allow us to prevent the first integrity attack mentioned above, where students could attempt to modify their rubric prior to submission.

1.3. Technique 2: Public Key Encrypted Submissions

(I) In conjunction with that, we will give each student i a unique MAC key K_i that only they and the professor (or whomever does the grading) will ever know. The professor will have a list of all students' MAC keys. Furthermore, the professor will have a public key / private key pair of PK, SK. Then, when every student is ready to submit their final *report*, they will use the Encrypt then MAC method to first encrypt their report as $C_r = E(report, PK)$ using the professor's public key PK and then generate their MAC using their MAC key K_i as $MAC = MAC(E(report, PK), K_i)$. The student will then finally submit their report ciphertext C_r along with their encrypted MAC MAC, resulting in the pair (C_r, MAC). This could also be combined with the above technique by having each student submit their report ciphertext, the encrypted report's MAC (which uses the student's unique key K_i) as well as their rubric and the rubric's MAC (which instead uses the staff only key K). Once the professor recieves these submissions, they can use the staff only key K to authenticate the submitted rubric. Then, they can use the student's MAC key K_i to authenticate their submitted ciphertext and they can use their private key SK to decrypt the student's report and get its plaintext version.

(II) This allows us to prevent against the confidentiality vulnerability of being able to read other students' unencrypted reports, while also fixing the authentication issue of making sure that reports all come from the right students.

2. Messaging with Public Keys

(a) Since Eve is just a passive eavesdropper, she can only see the encrypted ciphertexts sent, but this can still give her some vital information. By examining the ciphertext of a large number of messages, she'll be able to determine the relative frequency of different meeting places and times. She could further combine this with any existing knowledge she has (like the known frequency of their meeting spots/times) to fully decrypt certain messages.

(b) Since Eve knows Alice's public key PK_A, she's able to try encoding different meeting places and times using that public key until the result matches one of the ciphertexts she has seen in the chat. Once she gets a match, she'll permanently know what that ciphertext decrypts to, allowing her to start decoding full messages. She could even use an algorithm to run through all potential strings of a given length to see which of them match with the ciphetexts she's intercepted by eavesdropping. So now, instead of just knowing the relative frequency of different places/times, she'll be able to figure out what those places/times are through trial and error encrypting.

(c) I would try to add a random amount of characters before and after their actual message (maybe using pre-specified values so that I could remove them before showing them to the users) prior to encryption. This way, even the same messages won't always have the same ciphertext, so Eve won't be able to determine the frequency of different messages using just the ciphertexts or just brute force until she finds a message that encrypts to the ciphertext.

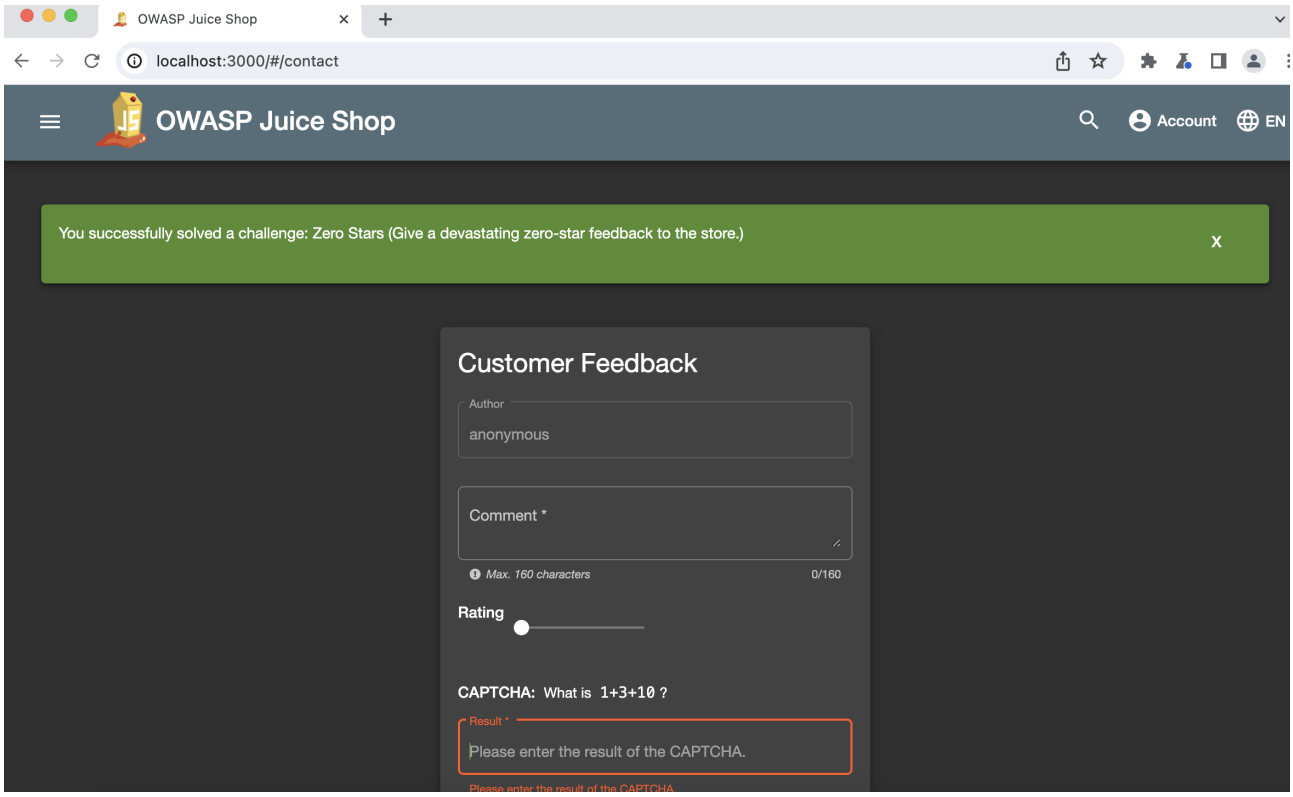
(d) **False.** Since this current protocol just uses public key cryptography, the public key needed to send a message to Alice is, by definition, public knowledge. Thus, anyone can send a message to Alice, and she has no guarantees that the person she is speaking to is actually who they say (eg. Bob). This is because public key cryptography on its own is almost solely focused on confidentiality rather than authenticity, making it very difficult for anyone but Alice to read her mail, but not doing anything about people trying to send messages pretending to be Bob.

3. Exceptional Access

(a) I largely agree with the thoughts of the authors of this article: I think that exceptional access can be an important tool for law enforcement agencies if used in a very targeted, lawful manner. However, the latter of those two adjectives is fairly difficult to define because, as the article states, who defines which "good"governments should be given the ability to request exceptional access? I think that this brings about a fundamental need for international law concerning this topic, where governing bodies together can decide which cases constitute a valid reason for requesting exceptional access, and which governing bodies have the authority to do so. Furthermore, I think that the primary reason I would support exceptional access in certain circumstances is the threat to life that an over-eager loyalty to privacy and the trust between user and developer could unintentionally cause, so I think that the public harm must be a key qualifying component for exceptional access.

(b) Personally, I don't find hacking to be a compelling mechanism for providing exceptional access, as it would be loosening restrictions on when that access could be granted and to whom. As this prompt mentioned, using hacking as a method for exceptional access encourages governments not only to not disclose security vulnerabilities to developers, but also to actively maintain their existence. While this most certainly does happen in real life, I find it to not be ideal as it incentivises governments to essentially compromise the security of a given application to maintain their exceptional access. And while some security vulnerabilities may be so small that only those with the resources of a full government behind them could exploit it, many will allow actors who would not normally be granted exceptional access the ability to compromise application security, greatly increasing the risk and privacy violations involved.

4. Burp Suite Lab



5. Better Passwords via a Browser Extension

(a) No, it unfortunately does not protect Alice against a dictionary attack, because attackers looking to get into a given website would be able to try various dictionary words, and use the extension to attempt a login with $hash([word]++[site])$ until they successfully log in. So this extension alone does not provide any additional protections to Alice's weak password. However, if the attackers didn't know Alice was using this extension it would actually prevent a dictionary attack from working because her password would end up being the random string produced by $hash(balloon++[site])$ rather than an actual word that would be attempted in a dictionary attack.

(b) No, Alice is not protected from a brute-force attack in this case. Assuming that the bank only stores the hashes of users' passwords and *bank.com*, attackers could unfortunately brute force through all potential passwords and try to send them to the site. This would run them through the browser extension which would calculate $hash([pwd] + +bank.com)$. The attacker could then use a tool to see what request is actually being sent to the *bank.com* server, and keep trying different passwords until they get one that hashes to the same value they stole from *bank.com*'s servers, unfortunately allowing them to retrieve her password.

(c) Yes, this actually does protect Alice from having her password stole, as we are unintentionally sending Eve $hash([pwd]++bank.co)$ but, due to the nature of hash functions, this will be a very different value from what the hash would be if Alice was trying to access any other website. Furthermore, the nature of our hash function would mean that it is infeasible to work backwards from a hash value to attain the input, so this would only be helpful if Eve wanted to log into *bank.co* as Alice (which wouldn't be very useful since it only belongs to her). The only exception to this would be if Eve was somehow able to use a similar brute force attack as in (b) to loop through all potential passwords and calculate $hash([pwd] + +bank.co)$ until she finds one that matches the value Alice sent her, allowing her to find the plaintext password and then calculate the hash for an arbitrary website.

(d) This extension increases the entropy of users' passwords because it converts their existing passwords (that will often have patterns or words) into a more random hash that will likely be a) longer, and b) less repetitive and pattern prone than a typical password, meaning it will be much more difficult to crack. This means the overall entropy of users' passwords increases.

Submitted by Student unknown *on 24 de febrero de 2024.*