Take-Home Midterm Solutions

Problem M-1. Protocol Design (Courtesy of Megumi Ando.)

The most overwhelmingly common (and disappointing) error we saw was in the fixes to the broken client-server authentication scheme. Almost everyone suggested some kind of ad-hoc fix to the protocol, such as:

- Making the nonce longer, so the plaintext spans more than one block.
- Doing a more complicated operation on the nonce (instead of just incrementing it).
- Forcing the client's IV to be a specific value, or disallowing certain values.
- Swapping the order of the (incremented) nonce and client identity.

The problem with all these solutions is that they ignore the core issue, which makes the attack possible in the first place: *encryption is/may be malleable — it is not the right tool for the job!* The authentication protocol is attempting to establish that the client knows the secret key K; however, as we have seen, it is possible to create a good-looking ciphertext (by mauling another one) without knowing what it means! Therefore, the proper thing to do in this scenario is to include a MAC: this establishes authenticity of the message, and in particular, a valid MAC cannot be generated without knowledge of K. To properly fix the bug, the client should also send a MAC of the response ciphertext. This ensures secrecy of the nonce n, as well as proper authentication of the client.

The following solution was submitted by Megumi Ando:

(a) If Alice sends Bob $(A, \{n_1\}_K)$, she expects Bob to send her something in the form $(B, \{n_2\}_K)$ to start another initiation, OR $(B, \{n_1 + 1\}_K)$ in response to her initiation. More specifically, if Alice sends Bob an encrypted nonce, and she receives the same nonce from Bob, she assumes that Bob is starting another initiation and sends Bob the correctly encrypted incremented nonce.

Let Mallory be a malicious adversary, who does not know K. She waits until Alice (who does knows K) sends her an encrypted nonce (A, C_1), where $C_1 = \{n\}_K$. Mallory then sends Alice the same encrypted nonce (M, C_1). Alice, who then assumes that Mallory is starting another initiation, correctly computes $C_2 = \{n + 1\}_K$ and sends Mallory the correctly encrypted incremented nonce (A, C_2). Mallory then sends Alice (M, C_2).

One way to fix this bug is to disallow two instances of the protocol from interleaving. That is, if Alice sends Bob $(A, \{n_1\}_K)$, she expects Bob to send her $(B, \{n_1 + 1\}_K)$ before he sends her $(B, \{n_2\}_K)$.

(b) The adversarial client (who does not know K) waits for the server to send two 128-bit blocks. Let IV be the first block. Let C be the second block.

Assume that the server correctly encrypts under AES in CBC mode. That is, $C = \text{AES}_K(\{S \circ n\} \oplus IV)$.

If the adversarial client guesses that n ends in a 0, (s)he sends to the server the following two 128-bit blocks:

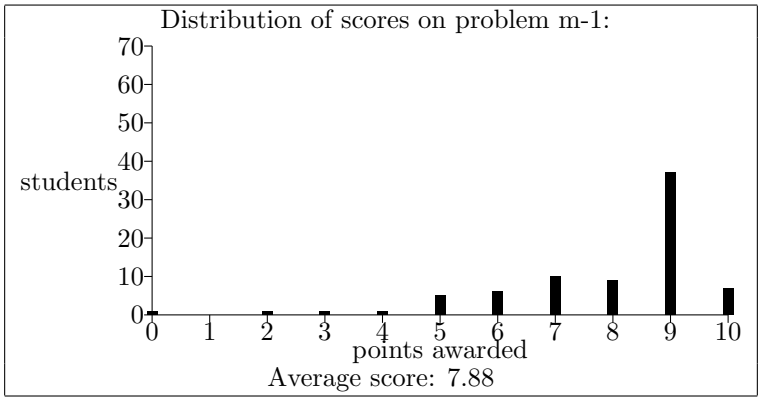
1. $C_0 = IV \oplus \{S \circ 0^{64}\} \oplus \{C \circ 0^{63} \circ 1\}$, AND
2. C.

To decrypt, the server does the following:

$$\begin{aligned} P &= \text{AES}_K^{-1}(C) \oplus C_0 \\ &= \text{AES}_K^{-1}(\text{AES}_K(\{S \circ n\} \oplus IV)) \oplus IV \oplus \{S \circ 0^{64}\} \oplus \{C \circ 0^{63} \oplus 1\} \\ &= \{S \circ n\} \oplus IV \oplus IV \oplus \{S \circ 0^{64}\} \oplus \{C \circ 0^{63} \circ 1\} \\ &= \{S \circ n\} \oplus \{S \circ 0^{64}\} \oplus \{C \circ 0^{63} \circ 1\} \\ &= \begin{cases} \{S \circ (n+1)\} & \text{if } n \text{ ended in a 0, OR} \\ \{S \circ (n-1)\} & \text{if } n \text{ ended in a 1.} \end{cases} \end{aligned}$$

Since n is randomly generated, the chance that it ends with a 0 is $\frac{1}{2}$. Thus, the adversarial client can authenticate itself with the server with probability $\frac{1}{2}$.

(c) The bug can be fixed by having the server check that the first block sent by client is the same first block IV that it sent to the client. (**TA note:** this is a somewhat OK answer, because it essentially acts as a home-grown MAC. Then again, re-use of an IV may compromise some secrecy of the nonce n. We'd prefer a well-studied, strong MAC, such as CBC-MAC under AES.)



Problem M-2. SSL Certs (Courtesy of Saad Shakhshir.)

This problem took an in-depth look at SSL certificates.

Full credit on part (a) required explaining that users trust Microsoft to provide the correct certificate, and that they trust that the cetificate is both valid now and valid in 2028. Ideally we wanted to see a discussion of how Moore's law will impact 1024-bit RSA and 160-bit SHA-1, although it was equally acceptable to say that the certificate would be revoked if it were compromised.

- One student noted that "it is unclear why VeriSign chose such a far-off expiration date." The reason that VeriSign did this is because it was such a tremendous pain in the late 1990s when we had to deal with certificates expiring in 1998 and 1999. This way, the logic goes, the computers using this certificate will be long-gone by the time it expires.
- Many students seemed confused between RSA bit length and AES bit length. These students implied that since an 80-bit key is considered secure today, and since computers are getting twice as fast every 1.5 years, then a 1024-bit key must be secure for at least $2^{\frac{1024-80}{1.5}}$ years, which is a really long time. But RSA keys are not cracked by doing a brute-force key search: they are cracked by factoring.

Full credit on part (b) required explaining that GeoTrust is verifying the abaility to receive mail and an email address within the domain, but is not verifying the business actually exists or is affiliated with Palm. Some students thought that GeoTrust was verifying the abaility to receive e-mail at `admin@palm.com` while others thought it was verifying the ability to receive email at `admin@store.palm.com`. Neither of your TAs have legal degrees and the GeoTrust CPS is kind of ambiguous on this matter, so either answer was accepted.

Full credit on part (c) required a discussion that we trust the people who run the MIT CA to be trustworthy, and we trust that the network will actually get us the correct CA key. So we are trusting DNS and the routers. Points were deducted for the students who stated that SSL is used to download the MIT CA key; it is not.

Full credit on part (d) required a discussion that SSL certificates prevents the web server from having to handle each user's username and Kerberos password. The disadvantage of the SSL certificate, though, is that they tend to be left on machines and are typically not guarded as well as passwords. Credit was also

accepted for other answers, but not for saying that SSL certificates prevents against eavesdropping in the network; regular SSL does that.

Here is an especially good answer for M-2 that was written by Saad Shakhshir.

(a) All Public Key Infrastructures (PKIs) start with at least one root certificate which is self-signed — that is, the owner and signer names are the same. Users of Internet Explorer (IE) can trust this certificate because of the reputation that the owner has for being trustworthy. The owner, in this case VeriSign, establishes this reputation by keeping secret the private key that corresponds to its public key. If this private key were stolen, then it could be used to falsely sign other web certificates. We believe that news of falsely signed web certificates would get out quite fast and given the nature of the Internet and the swiftness in which information could spread, this news would travel quickly to the places where fixes could be generated and this specific root certificate could be revoked.

To verify that we do in fact have the appropriate VeriSign certificate, we can look at its public key and its thumbprint. These can be checked against the values that VeriSign has posted on its website. Microsoft could have written IE such that when someone wants to check that specific VeriSign website, the user is actually diverted to some other page (either locally or somewhere else) that holds a page looking just like the VeriSign page only with a fake public key matching the fake one installed on IE. However there are other websites that display VeriSign's public key information and one could use another program/operating system to check. There would have to be a very large and coordinated conspiracy to have IE come with a fake root certificate. Although most people do not go through the trouble of verifying the root certificates that come installed on IE and it might not be hard to fool the average user. However, if word of such an attempt got out, then Microsoft's reputation would be severely damaged. We believe that it would not be in their interest to attempt this.

The length of the public key is 1024 bits. This is considered by RSA to be secure for the "immediate future". Currently the largest key to be broken is one of 512 bits. It is predicted that 768 bits may be broken in the next few years. Clearly one cannot predict exactly when a key is going to be broken. nNevertheless, if we use historical data to predict when a 1024-bit key may be expected to be factored, then an estimate obtained from the RSA website is around 2037. There is an alternative theoretical model that leads to a prediction of 2018, also from the RSA website. The discrepancy can be attributed to how much effort is being put into actually breaking these keys. Given these two estimates, it would be appropriate to say that by the year 2028, VeriSign's key will no longer be secure. 2048 bit keys are already being used and in the future VeriSign will have to increase the length of its key (or use another cryptographic method) if it wishes to remain a trusted root certificate.

(b) This certificate is an SSL certificate and is not the same as the VeriSign certificate, which is a root certificate. Recall that a root certificate is issued to and signed by the same entity. This certificate was issued by Equifax Geotrust to store.palm.com.

The value of SSL is protected by the strength of a standard two-point validation process [1]:

1. Verify that the applicant owns, or has legal right to use, the domain name featured in the application.
2. Verify that the applicant is a legitimate and legally accountable entity.

Thus GeoTrust, by stating that the organization is not validated, has not performed step 2 of the above process. When a person registers a domain name, any ownership information can be given in the application. This information is never verified. Hence this certificate only tells us that that the applicant of the certificate is authorized to use that domain name, it does not tell us anything about the nature of the applicant. We could be relying on potentially untrustworthy information.

To get a certificate for store.palm.com, the owner of that domain must do the following: "Subscribers submit their Public Key to GeoTrust for certification electronically through the use of a PKCS#10 Certificate Signing Request (CSR) or other package digitally signed by the Subscriber?s Private Key

[1] http://www.instantssl.com/ssl-certificate-support/guides/ssl-certificate-validation.html

in a session secured by Secure Sockets Layer (SSL). At a minimum, the Subscriber must provide the following data in or with the CSR: Common Name, Organization, and Country. The following additional information is required on the enrollment form: the names, e-mail addresses, and telephone numbers for the Administrative, Technical, Support, and Billing points of contact [2]."

Once this has been completed, GeoTrust will send an email to an email address at that domain name (this might be one of the emails specified in the application) authorizing the certificate request. Once a confirmation has been received, it will issue the certificate.

To get an Equifax Geotrust certificate for a computer in the `mit.edu` domain, we found this website `https://www.webii.net/support/sslcertificate.html`. All we would need to do is fill in that form, pay the required amount, and we would be able to obtain the certificate. The only way they verify our identity is by the email address and so we must specify an account that we have access to. We just need to make sure that the information we input fulfills their requirements. For example, it says that the common name (the domain name) must be registered to the organization that is specified in the organization field. We can do a simple `whois` query on the computer that we are attempting to get a certificate for and make sure that the organization specified in the domain name registration is the same as the one we give in the form. In this case it would probably just be MIT.

(c) We are trusting that the domain name `http://bs.mit.edu/mitca.ca` will resolve to the IP address of the appropriate MIT computer. Thus one attack would be to intercept the DNS request and send back another IP address. The attacker could then trick us into communicating with a different computer and downloading a different certificate.

Another mechanism we are trusting is that of the browser. Assuming that we download the appropriate certificate from the MIT server, we believe that the browser is in fact installing that particular certificate. If the browser were written maliciously, it could install a different certificate other than the one we downloaded. This could lead us to authenticate malicious websites.

(d) One advantage is that once a student obtains a client certificate, she no longer has to re-enter her username and password. If there is any risk that this user/pass combo gets stolen, then it is reduced by enabling the student to enter it only once. Afterwards, she has this digital certificate that authenticates her with the Stellar and MIT Libraries and that is stored locally on her computer. So it is not sent over the network every time she authenticates and there is less of a risk of it getting stolen.

A disadvantage is that on IE, there is an option that tells you how much security you want on this client certificate when you get it. If you select 'medium' as your setting (which is the default), then it does not prompt you for a password every time you use your certificate to authenticate yourself. Thus anyone with access to your computer can authenticate themselves as being you. An adversary could also export your certificate and then use it from any other computer to authenticate themselves as you. If the servers prompted for a username and password every time (or if you select the 'high' setting and put a password), then this would not be possible.

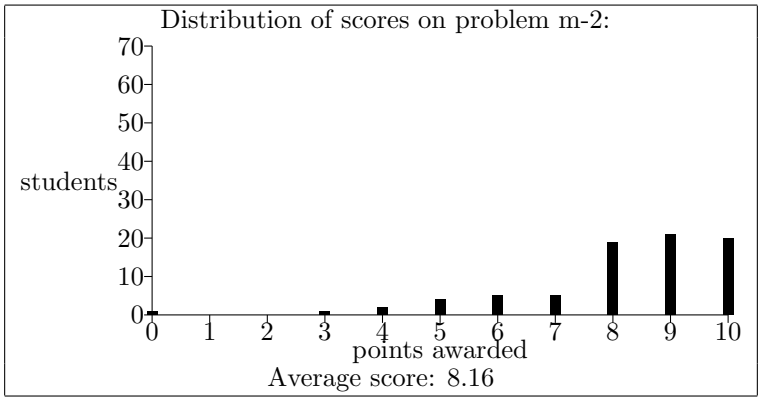
References

1. "Taking The Confusion Out Of Digital Certificates" (http://www.networkcomputing.com/909/909colmoskowitz.html)
2. "Circuits for Integer Factorization: A Proposal" (http://cr.yp.to/papers/nfscircuit.ps)
3. "Schneier.com Crypto-Gram March 15, 2002" (http://www.schneier.com/crypto-gram-0203.html#6)
4. "Managing the Digital Enterprise Security Key Length vs. Cryptographic Strength" (http://digitalenterprise.org/security/key_length.html)
5. "RSA Laboratories: A Cost-Based Security Analysis of Symmetric and Asymmetric Key Lengths" (http://www.rsasecurity.com/rsalabs/bulletins/bulletin13.html)

[2] http://www.geotrust.com/quickssl/cps (c)01

6. "RSA Laboratories Cryptography FAQ How large a key should be used in the RSA cryptosystem" (http://www.rsasecurity.com/rsalabs/faq/3-1-5.html)

7. "SSL Certificate Free SSL Secure Server Certificate Branded SSL" (http://www.instantssl.com/ssl-certificate-support/guides/ssl-certificate-validation.html)



Problem M-3. Patch Management

Grading: Because this was such an easy problem, we expected good, detailed answers. We awarded 3 points for correct answers in (a), (b) and (c), 1 point for good writing. 1 point deducted for each annoying technical error.

Part (a): It's important to validate the updates so that the user doesn't accidentally download booby-trapped updates! Practically every student got this part correct. But it's also important to authenticate the website — if the user went to the wrong website, that website might say "there are no updates today!" when in fact there is an important update on the correct website! Or the hostile website might simply send old updates — updates with known problems. Surprisingly, most students in the class didn't get this half of the answer.

One student had came up with a truly important reason to authenticate the server, as opposed to the updates: so that the user doesn't pay the wrong company! This answer received full credit.

Part (b): It's important to authenticate the user's computer to the remote website to make sure that the user is licensed to have the software that they will be receiving. (Another reason to authenticate the website in part a above is that some pirate website might not enforce MSC's software licensing system.)

Quite a few students noted that another reason to authenticate desktops is so that MSC can keep track of which desktops are updated and which aren't. Yet another reason to authenticate desktops is to prevent denial-of-service attacks that might happen if thousands of desktops repeatedly ask for the same update: with strong authentication, these desktops can be locked out.

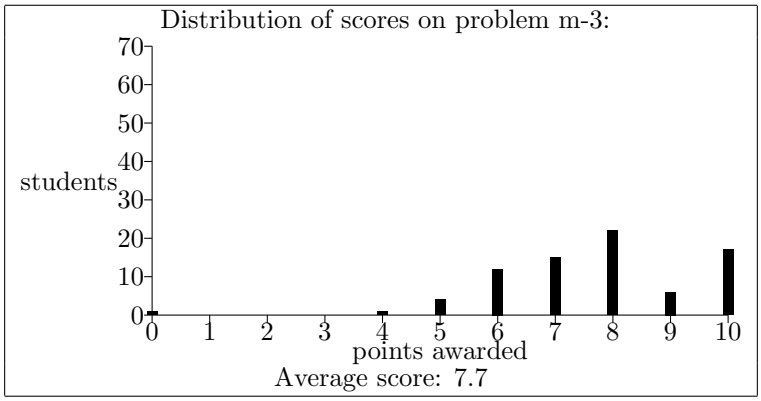
Gerardo Viedma came up with a clever attack that's possible if you don't authenticate the client: "If the individual desktop computers were not authenticated, then it would be possible for an attacker to maliciously craft a software update request for a different desktop computer than itself. If the attacked desktop computer were to accept the authenticated update received from MSC but which it did not in fact request, then the attacker could use this method to selectively update the software of the target system. If the attacker knows enough about the attacked system (for example, the software that it is running), it could request a specific update to be sent to the client that the malicious desktop knows will create a vulnerability on the target host."

Part (c): MSC doesn't want to be bothered for every update that BGC distributes, but MSC still wants to charge BGC for using the online patch service! So MSC puts on its webserver a list of authorized software partners. For each one is has a certificate with that partner's public key, signed by MSC, and the URL of the partner's update server. The PC goes to the MSC website, downloads the list of approved partners,

and sees if any software is installed on the computer that has the same publisher ID as one of the approved partners. If such software i installed, the PC goes to the partner's website and downloads the appropriate patches.

Common errors:

- Many students simply described how MSC could set up a PKI without describing how the update software would use this PKI to find the BGC server and updates. These students had between 1 and 2 points deducted, depending on how much work they did on the PKI.
- Many students confused the names MSC and BGC, or switched to other initials half-way through their essays. These students had 1 or 2 points deducted, depending on the severity of their errors.



Problem M-4. Sharing Secrets with Deities (Courtesy of Ioan Tudor Leu.)

The most common error on this problem was using a value of p "that is a prime approximately equal to $b/(t-m)$." In fact, p should have *length* about $b/(t-m)$, so it's *value* is about $2^{b/(t-m)}$!

Another pet peeve of the grader was encountered far too frequently: many solutions said "use a polynomial of *order* $t-1$," when they meant *degree*. The order of a mathematical object v is the smallest positive exponent x for which v^x equals the identity element; the degree of a polynomial $f(x)$ is the largest degree of all its monomials (i.e., the largest exponent of x appearing in the polynomial).

The following very clean solution was submitted by Ioan Tudor Leu:

(a) Divide the secret s into $t-m$ parts $a_0, a_1, \ldots, a_{t-m-1}$, each of length $\frac{b}{t-m}$. Also, generate randomly m other terms, $a_{t-m}, \ldots, a_{t-1}$.

Take then the $t-1$ degree polynomial $q(x) = a_0 + a_1 x + \ldots + a_{t-1}x^{t-1}$ and distribute to each god i a share of the secret $D_i = q(i)$, of length $\frac{b}{t-m}$. All arithmetic is done modulo a prime p of length $\frac{b}{t-m}$.

(b) Any t shares of the secret are in fact t values of the polynomial for t different arguments, and they completely determine the polynomial. There are in total t equations of the type $D_i = q(i)$ and they are sufficient to determine the t coefficients of the polynomial, of which the first $t-m$ represent the secret.

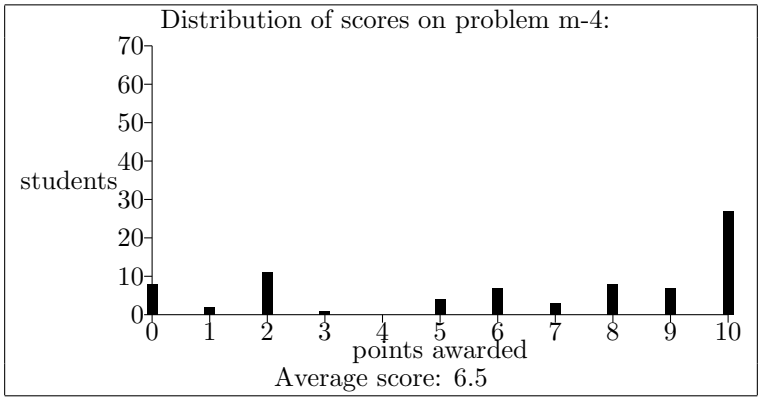
(c) As we said in (b), each share of the secret $D_i = q(i)$ can be interpreted as an equation with the unknowns $a_0, a_1, \ldots, a_{t-1}$:

$$a_0 + ia_1 + i^2 a + 2 + \ldots + i^{t-1}a_{t-1} = D_i$$

Suppose we have $k < t$ shares of the secret, therefore k equations. The solution to the system of equations is described by $t-k$ independent variables and k equations which determine the value of the other k unknowns as a function of the independent variables.

In the case of $k \leq m$ shares, at least $t-m$ variables are independent, so we can choose any values for the first $t-m$ coefficients of the polynomial, and still satisfy all k equations with appropriate choices for the rest of the coefficients. This means that no information is divulged about the secret, since there is no constraint on the coefficients $a_0, a_1, \ldots, a_{t-m-1}$, which represent the secret.

When $k > m$, however, at least one of the first $t-m$ coefficients is dependent on the others. This means that the group of k deities can exclude from consideration certain sets of values for the first $t-m$ coefficients, so there is a leak of information in this case.



Problem M-5. Authenticated Key Exchange (Courtesy of Oana Stamatoiu.)

The most common error we saw was in the answers to the forward-secrecy question. Forward secrecy is *not* about whether the compromise of one session key affects the security of another session; it *is* about whether the compromise of *long-term* secrets (such as private keys) affects the security of prior recorded sessions. Many students either did not understand this concept, or claimed that "because r_A and r_B add randomness to the protocol," even with the private keys x_A, x_B, an adversary could not decrypt old sessions. Adding randomness is not enough — one must analyze the protocol with the assumption that the adversary will learn (or already knows) the long-term secrets (but not the ephemeral ones).

Another common quasi-error was in the self-authentication attack. Many students said that the adversary should merely return $t_A^{-1} \pmod p$, which causes K to be 1. This is a (mostly) acceptable answer, however it is easy for Alice to take countermeasures against it, because the attack is deterministic. We reserved full credit for those attacks which were sufficiently random as to prevent any hope of detection.

The following solution was submitted by Oana Stamatoiu:

(a) Alice computes:

$$K = (g^{r_B})^{x_A} \times (g^{x_B})^{r_A} = (t_B)^{x_A} \times (y_B)^{r_A} \tag{1}$$

Alice knows x_A (her secret key) and r_A (which she chooses in step 2). She also knows y_B (Bob's public key) and t_B (which she gets from Bob in step 3). Thus, she can compute $K = (t_B)^{x_A} \times (y_B)^{r_A}$

Bob computes:

$$K = (g^{x_A})^{r_B} \times (g^{r_A})^{x_B} = (y_A)^{r_B} \times (t_A)^{x_B} \tag{2}$$

Bob knows x_B (his secret key) and r_B (which he chooses in step 3). He also knows y_A (Alice's public key) and t_A (which he gets from Alice in step 2). Thus, he can computer $K = (y_A)^{r_B} \times (t_A)^{x_B}$

(b) In step 3, Bob sends Alice $t_B = \frac{g^{r_B}}{g^{r_A}} = \frac{g^{r_B}}{t_A}$

Alice can compute the key K just like before. All Bob has to do is compute $K = (y_A)^{r_B}$.

Proof:

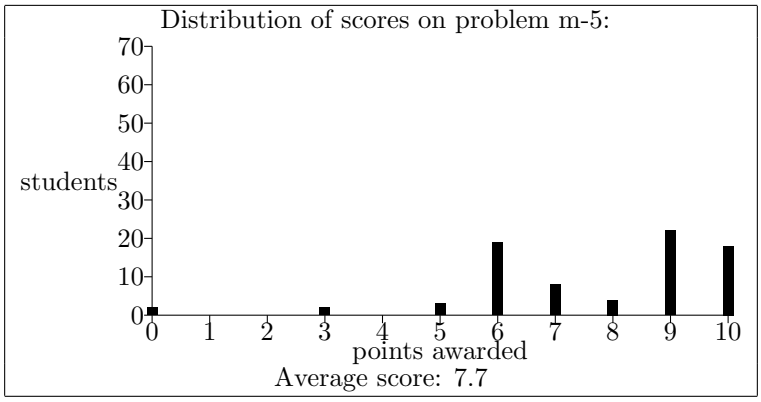
$$\begin{aligned}
(y_A)^{r_B} &= g^{x_A \times r_B} \\
&= g^{x_A \times r_A - x_A \times r_A + x_A \times r_B} \\
&= (g^{x_A \times r_B - x_A \times r_A}) \times g^{x_A \times r_A} \\
&= (g^{r_B - r_A})^{x_A} \times g^{x_A \times r_A} \\
&= (\frac{g^{r_B}}{g^{r_A}})^{x_A} \times y_A^{r_A} \\
&= (t_B)^{x_A} \times y_A^{r_A} \\
&= K
\end{aligned}$$

This value of K is the same as the one computed by Alice.

(c) A protocol is forward secure if the compromise of a long term secret (such as x_A and x_A) at some point in the future, does not compromise the security of communications made in the past using that secret. In our case, we have

$$K = g^{x_A \times r_B} \times g^{r_A \times x_B} = t_B^{x_A} \times t_A^{x_B} \tag{3}$$

We can see that this protocol is not forward secure: if Eve finds both x_A and x_B at some point in the future, *and* if she also has the transcripts of the communications between Alice and Bob (which contain t_A and t_B, given out in steps 2 and 3), then Eve can easily reconstruct K, as in equation (3) above.



Problem M-6. Academic Honesty [3 points]

This was the easiest question on the midterm: everyone who turned in an answer got full credit. Thank you for keeping your academic integrity!