CS 161 Computer Security
Spring 2010 Paxson/Wagner

MT2 Solns

Midterm 2 exam solutions

Please— do not read or discuss these solutions in the exam room while others are still taking the exam.

Problem 1. [True or false] (14 points)

(a) [TRUE] or FALSE: If Alice has a message to send to Bob and she wants to encrypt the message using asymmetric cryptography so that no one other than Bob can read it, she does so by using Bob's public key.

(b) [TRUE] or FALSE: SSL and TLS provide essentially the same end-to-end security properties.

(c) TRUE or [FALSE]: Properly used, a MAC provides both confidentiality and integrity.

(d) TRUE or [FALSE]: DNSSEC uses SSL between different name servers to certify that the results of DNS queries match those that the name servers are authorized to provide.

(e) [TRUE] or FALSE: In the United States, if a company posts a privacy policy on their web site and fails to comply with it, they can be prosecuted for false advertising.

(f) TRUE or [FALSE]: An attraction of public key cryptography is that, if implemented properly, the algorithms generally run much faster than those for symmetric key cryptography.

(g) TRUE or [FALSE]: Memory protection, as found in a typical operating system, prevents malicious code running in kernel mode from writing to application-owned pages.

Problem 2. [Multiple choice] (18 points)

(a) TLS uses the following cryptographic techniques:
[(i) Asymmetric-key cryptography.]
[(ii) Symmetric-key cryptography.]
[(iii) Cryptographic hash functions.]
[(iv) PKI certificates.]
[(v) Nonces.]
(vi) None of the above.

(b) Which of the following properties must a cryptographic hash function provide?
(i) Key revocation.
[(ii) Collision resistance.]
[(iii) A deterministic mapping from input to output.]
(iv) One-to-one mapping of input to output.
[(v) Difficulty of finding an input that matches a given hash.]
(vi) None of the above.

(c) What risks arise when using the same key to encrypt both directions of a communication channel, that aren't present if using different keys for the different directions?
(i) Message tampering by flipping bits in the ciphertext.
[(ii) Reflection attacks.]
(iii) Hash collisions.
(iv) Eavesdropping attacks.
(v) Denial-of-service.
(vi) None of the above.

(d) As we saw in class, WEP is vulnerable to active attacks that allow an active attacker to flip bits in the ciphertext and thereby cause unauthorized modifications to the message received by the recipient. What would be the best defense against this kind of attack?
(i) Use a different key for each direction and for each wireless device.
(ii) Protect the ciphertext using a MAC.
(iii) Encrypt using AES in Cipher Block Chaining (CBC) mode.
(iv) Encrypt using AES in Electronic Code Book (ECB) mode.
(v) Prepend a random 32-bit nonce to the packet before applying the CRC and encrypting it.

Problem 3. [Terminology] (14 points)

1. The security goal of ensuring that a communication arrives at the recipient in a form identical to what the sender transmitted. **Answer:** Message integrity.

2. A widely used, standardized symmetric key encryption algorithm. **Answer:** AES.

3. A way of checking whether the private key matching the public key in a certificate has been compromised and so the certificate should no longer be accepted. **Answer:** Revocation list.

4. A symmetric-key algorithm for ensuring that a message has not been tampered with. **Answer:** MAC.

5. The amount of uncertainty that an attacker faces when trying to guess an unseen value. **Answer:** Entropy.

6. An approach by which users can build up a degree of confidence in a public key's validity without requiring a trusted root of authority. **Answer:** Web-of-trust.

7. An algorithm for digitally signing data with a private key such that anyone with possession of the corresponding public key can verify the signature. **Answer:** RSA.

8. A signed statement by a trusted authority that a given public key indeed belongs to a given party. **Answer:** Certificate.

9. A value used in symmetric key cryptography to ensure that a new session that transmits the same text as a previous session does not result in identical ciphertext. **Answer:** IV. (Nonce is also acceptable.)

10. A way of constructing a stream cipher, given a block cipher. **Answer:** Counter mode.

11. The notion that the security of a well-designed cryptography algorithm should not rely upon the secrecy of the algorithm itself but only on the secret keys it uses. **Answer:** Kirchoff's principle.

12. A widely used, standardized cryptographic hash function. **Answer:** SHA256.

13. A Unix operating system mechanism that enables a program to execute with the privileges of a different user identity rather than the identity of the user who invoked the progam. **Answer:** Setuid.

14. A trusted third party who provides a way for one party to learn the public key of another party. Web browsers have a list of these trusted third parties, to support communication using HTTPS. **Answer:** Certificate authority.

Problem 4. [Cryptography] (15 points)

(a) SuperMail wants every email to be authenticated and protected from modification or tampering while it is transit from the sender to the receiver. Suppose Alice is sending an email M to Bob. Given SuperMail's design constraints, which of the following options would be a secure way to protect the authenticity and integrity of her email?

(i) Alice's software should encrypt M under Bob's public key. In other words, Alice's software should send $E_{K_B}(M)$ to Bob.

Comment: Encryption does not provide authenticity/integrity. Anyone can send such a ciphertext.

(ii) Alice's software should send M along with a digital signature on M using Alice's private key. In other words, Alice should send $M, \text{Sign}_{K_A^{-1}}(M)$.

(iii) Alice's software should choose a new symmetric key k for this email, send an encryption of k under Bob's public key, and also send an encryption of M under k using a stream cipher such as RC4. In other words, Alice should send $E_{K_B}(k), M \oplus \text{RC4}(k)$.

Comment: Encryption does not provide authenticity/integrity. Anyone can send such a ciphertext. Also, it's vulnerable to message modification (flip bits in the second part of the packet).

(iv) Alice's software should choose a new symmetric key k for this email, send an encryption of k under Bob's public key, and also send an encryption of M under k using AES in CBC mode. In other words, Alice should send $E_{K_B}(k), \text{AES-CBC-Encrypt}_k(M)$.

Comment: Encryption does not provide authenticity/integrity. Anyone can send such a ciphertext. Also, it is vulnerable to message modification: modifying the last 128 bits of the packet disrupts only the last 128 bits of the message that Bob receives, and Bob won't detect that the message was tampered with. This violates the message integrity goal.

(v) Alice's software should choose a new symmetric key k for this email. Then it should send four pieces of information: the message M, a MAC on M under the key k, an encryption of k under Bob's public key, and a digital signature on k using Alice's private key. In other words, Alice should send $M, \text{MAC}_k(M), E_{K_B}(k), \text{Sign}_{K_A^{-1}}(k)$.

Comment: Once Bob receives one such message, he can send forged messages to Carol and make Carol think Alice sent them. For instance, Bob can send $M', \text{MAC}_k(M'), E_{K_C}(k), \text{Sign}_{K_A^{-1}}(k)$ to Carol, and Carol will accept this thinking it came from Alice.

Also, the signature on k might reveal the value of k. Digital signature schemes are not guaranteed to provide confidentiality protection for the message that was signed. (In some signature schemes, the signature reveals the message that was signed; in others, it does not. Both possibilities are allowed by the definition of security for digital signatures.) Consequently, a man-in-the-middle might be able to recover k from $\text{Sign}_{K_A^{-1}}(k)$ and then modify M and recompute a new MAC that will be valid (using his knowledge of k).

(b) Let M be a confidential email that Alice wants to send to Bob, K_B be Bob's encryption public key, and K_A^{-1} be Alice's private key for signing. Which of the following options would be the best choice for protecting confidential emails?

(i) Send $E_{K_B}(M), \text{Sign}_{K_A^{-1}}(K_B)$.

Comment: This does not provide message integrity/authentication. A man-in-the-middle can change $E_{K_B}(M)$ to $E_{K_B}(M')$.

(ii) Send $E_{K_B}(M), \text{Sign}_{K_A^{-1}}(M)$.

Comment: May fail to provide message confidentiality, since $\text{Sign}_{K_A^{-1}}(M)$ might reveal M. Also, if Alice sends the same message twice, the signature will be the same both times (in some signature schemes, at least), which might be an undesirable violation of confidentiality.

(iii) Send $E_{K_B}(M), \text{Sign}_{K_A^{-1}}(E_{K_B}(M))$.

(iv) Send $E_{K_B}(k), \text{Sign}_{K_A^{-1}}(E_{K_B}(k)), E_k(M)$ where k is a new symmetric key chosen for this email and E_k represents encryption under k with a symmetric-key encryption algorithm.

Comment: Does not provide message integrity/authentication. For instance, if the symmetric-key encryption algorithm chosen is a stream cipher, then an attacker can flip bits in $E_k(M)$. (Also, it might be vulnerable to "reaction attacks" where a man-in-the-middle attacker modifies $E_k(M)$ and then observes Bob's reaction, to try to learn something about M.)

(v) Send $E_{K_B}(k), \text{Sign}_{K_A^{-1}}(E_{K_B}(k)), E_k(M), \text{MAC}_k(M)$ where k is a new symmetric key chosen for this email, E_k represents encryption under k with a symmetric-key encryption algorithm, and MAC_k represents a message authentication code (MAC) using key k.

Comment: Might not provide message confidentiality. Depending upon the MAC algorithm chosen, the $\text{MAC}_k(M)$ might reveal partial (or complete) information about M. Also, using the same key k for both encryption and MACing is poor practice.

Problem 5. [Local system security and privacy] (21 points)

(a) Suppose you are concerned that your browser has malicious code running within it, though you are confident that your operating system has not been compromised. You type `www.twitter.com` into your browser's address bar to take you to the Twitter site. Are there steps you could take (which could involve additional effort on your part) to check whether your browser sent any information to `www.twitter.com` via cookies as part of that request?

(i) Yes. (ii) No.

Justification 1: You can run a sniffer (e.g., Wireshark) to observe the network traffic actually sent by your browser, and check whether it contains any cookies.

Justification 2: You could configure your browser to route your all HTTP requests through a web proxy you've written that checks whether the request includes any cookies.

(b) Now consider a slightly modified situation where you are confident that your browser has not been tampered with, but you are concerned that your operating system may have been compromised.

You type `https://secrets.cs.berkeley.edu` into the browser's address bar, and your browser establishes a TLS connection to `https://secrets.cs.berkeley.edu`. That web server responds with a Web form for you to type in your username and password, and your browser sends back your answers via TLS.

Can malware running inside the operating system extract your username and password?

(i) Yes. (ii) No.

Justification 1: The malware could read all your keystrokes before it passes them on to the browser.

Justification 2: The malware could read the username and password out of your browser's address space.

Justification 3: The malware could act as a man-in-the-middle, so you are actually communicating with the attacker. The attacker would have to provide a bogus certificate, but the malware could prevent

detection of the attack by tampering with the results of file read syscalls for the file where the browser stores trusted certificates.

(c) Suppose now that you are confident that both your browser and your operating system have not been tampered with, but you believe a user-level process that runs as user `daemon` has been infected and is running malware. Permissions on your system allow user `daemon` to read your files (which includes the browser executable) but not to write them. You are confident that your OS correctly implements these permissions and provides process-level isolation and memory protection. However, user `daemon` is allowed by the operating system to "sniff" packets received or sent by your system's network interface card. Assume that your browser is free of bugs.

You again make a TLS connection to `https://secrets.cs.berkeley.edu`. Can the malware running as user `daemon` extract your username and password?
(i) Yes. (ii) [No.]

Justification 1: The username and password will be encrypted, so the malware can't read them by sniffing network packets.

Justification 2: TLS provides end-to-end security, so the malware can't use the sniffing interface to recover the password for the same reason that an eavesdropper can't.

Problem 6. [An alternate authentication scheme] (18 points)

(a) [ACCURATE] or INACCURATE: The Gingerbread scheme provides better security against phishing than password-based authentication: in the Gingerbread scheme, since users don't know their own secret authenticator A_U, users can't be easily fooled into typing their authenticator into a phishing web site.

(b) ACCURATE or [INACCURATE]: The Gingerbread scheme has a serious security flaw: since every web site can read all cookies stored in the browser, if the user visits a malicious third-party web site, the malicious web site could learn the user's authenticator A_U and then impersonate the user and access the user's bank account.

Comment: Third-party web sites cannot read cookies set by `www.bankobytes.com`; the browser's same-origin policy prevents that.

(c) ACCURATE or [INACCURATE]: The Gingerbread scheme has a serious security flaw: it can be easily broken by an attacker who simply tries to connect to the bank's web server many times, trying a different guess at A_U each time.

Comment: There are 2^{128} possible values for A_U, all equally likely, so an attacker will have to try all of them (or, on average, at least around half of them). If an attacker can make 1000 requests per second, it would take the attacker about 2^{118} seconds to try all possible 128-bit values. 2^{118} seconds is about 2^{93} years. This attack is unlikely to succeed within our lifetime, to say the least.

(d) [ACCURATE] or INACCURATE: One shortcoming of the Gingerbread scheme is that if the user leaves their computer logged in, then others who have physical access to the user's computer (e.g., the user's family members or the user's roommates) can impersonate the user and obtain access to the user's bank account.

Comment: In practice, a bank might deal with this by requiring users to authenticate with both a password and a Gingerbread cookie. The password can be stolen by phishers, but that's OK; phishers won't be able to steal the authenticator. The authenticator can be misused by roommates and family

members, but that's OK; they probably won't be able to guess the user's password, and we might not be terribly worried about people mounting a phishing attack against their own roommate.

(e) ACCURATE or INACCURATE: One feature of the Gingerbread scheme is that if the user wants to visit their bank website, but forgets to use the bookmark and types in `http://www.bankobytes.com` into the browser address bar, then this mistake will not compromise the user's security. From the standpoint of security, this is a good human factors engineering, because it means that this kind of user error does not cause a serious security breach.

Comment: If the user types in `http://www.bankobytes.com`, the user will get an error page, and the cookie will not be transmitted. Thus, the Gingerbread scheme fails safe in this situation.

Also, if the user mis-types the bank address and types in, say, `http://www.bnkobytes.com`, we're still OK, even if that other site is controlled by an attacker (a malicious typo squatter). The authenticator is stored in a secure cookie, so it will not be sent over any HTTP connection and thus will not be revealed to the attacker in this example.

(f) ACCURATE or INACCURATE: A potential security risk in the Gingerbread scheme is that, if Gingerbread's scheme is widely adopted by many banks, attackers might start trying to attack the process by which users initially sign up for online banking, instead of trying to steal the user's password.

Comment: This is indeed a risk.

A similar issue is that banks will probably need some way to handle situations where users clear all their cookies, install a new browser, or buy a new computer. Banks might provide a special sign-up protocol so the user can register their browser in these situations; but this creates an opportunity for phishers to set up a phishing site that tries to mimic the special sign-up protocol.

(g) ACCURATE or INACCURATE: A potential privacy risk in the Gingerbread scheme is that, if Gingerbread's scheme is widely adopted by many banks, advertising networks could use the cookie set by the bank to track users as they browse third-party web sites, without the consent of the user or the user's bank.

Comment: The same-origin policy will not allow third-party web sites or advertising sites to read or receive the cookie set by `http://www.bankobytes.com`, so they cannot use this cookie to track users (without the cooperation of BankOBytes).

(h) ACCURATE or INACCURATE: The only benefit of using HTTPS connections (instead of HTTP) in the Gingerbread scheme is ensuring that the user's browser is talking to the correct bank and not an imposter.

Comment: It also prevents eavesdroppers from learning the authenticator.

(i) ACCURATE or INACCURATE: If the bank preferred to minimize the amount of data that needs to be stored on the bank web server, there is an alternative that is also secure but needs less state on the bank web server: instead of making A_U a random 128-bit value, the bank could form A_U as the concatenation of the user's account number N_U and a MAC on the account number under some key k stored on the bank web server. In other words, in the alternate scheme, the bank web server stores a single key k (never revealed to anyone and the same for all users), and when a user signs up for online banking, the bank sets a cookie containing $A_U = (N_U, \text{MAC}_k(N_U))$ on the user's browser. This alternative is also secure and avoids the need to store the association (N_U, A_U) on the bank server.

Comment: This is basically the "SYN cookie" idea, applied to this context.