Preparing for the Final

MATH 481/581 — Cryptography
Wednesday, May 3

Please print your name:

Problem 1. The final exam will be comprehensive, that is, it will cover the material of the whole semester.

(a) Do the practice problems for both midterms.

(b) Retake all quizzes and both midterms (posted with and without solutions).

(c) Do the problems below. (Solutions will be posted soon.)

Bonus challenge. Let me know about any typos you spot in the lecture sketches or the posted solutions (surely, there should be some). Any typo, that is not yet fixed on our course website by the time you send it to me, is worth a small bonus.

Problem 2. Consider a block cipher with 5 bit block size and 5 bit key size such that

$$E_k(b_1b_2b_3b_4b_5) = (b_2b_5b_4b_3b_1) \oplus k.$$

(a) Encrypt $m = (010101010101010\ \ldots)_2$ using $k = (10001)_2$ and ECB mode.

(b) Encrypt $m = (010101010101010\ \ldots)_2$ using $k = (10001)_2$ and CBC mode ($\text{IV} = (10011)_2$).

Solution. $m = m_1m_2m_3\ldots$ with $m_1 = 01010$, $m_2 = 10101$ and $m_3 = 01010$.

(a) $c_1 = E_k(m_1) = 10100 \oplus 10001 = 00101$

$c_2 = E_k(m_2) = 01011 \oplus 10001 = 11010$

Since $m_3 = m_1$, we have $c_3 = c_1$. Hence, the ciphertext is $c = c_1c_2c_3\ldots = (00101\ 11010\ 00101\ \ldots)$.

(b) $c_0 = 10011$

$c_1 = E_k(m_1 \oplus c_0) = E_k(01010 \oplus 10011) = E_k(11001) = 11001 \oplus 10001 = 01000$

$c_2 = E_k(m_2 \oplus c_1) = E_k(10101 \oplus 01000) = E_k(11101) = 11011 \oplus 10001 = 01010$

$c_3 = E_k(m_3 \oplus c_2) = E_k(01010 \oplus 01010) = E_k(00000) = 00000 \oplus 10001 = 10001$

Hence, the ciphertext is $c = c_0c_1c_2c_3\ldots = (10011\ 01000\ 01010\ 10001\ \ldots)$. □

Problem 3. We use the silly hash function $H(x) = x \pmod{25}$.

Alice's public RSA key is $(N, e) = (55, 13)$, her private key is $d = 17$.

(a) How does Alice sign the message $m = 3141592$?

(b) How does Bob verify her message?

(c) Verify whether the message $(m, s) = (1234, 9)$ was signed by Alice.

(d) Give an example of a collision of our hash function.

Armin Straub
straub@southalabama.edu

Solution.

(a) $H(m) = 17$. The signature therefore is $s = H(m)^d = 17^{17} \pmod{55}$.

Doing the usual binary exponentiation ($17^2 \equiv 14$, $17^4 \equiv 31$, $17^8 \equiv 26$, $17^{16} \equiv 16$), $17^{17} = 17^{16} \cdot 17 \equiv 52 \pmod{55}$. Hence, the signature is $s = 52$.

(b) Bob receives the signed message $(m, s) = (3141592, 52)$.

He computes $H(m) = 17$ and then checks using the public key whether $H(m) \equiv s^{13} \pmod{55}$. Indeed, doing the usual binary exponentiation, $52^{13} \equiv 17 \pmod{55}$, so the signature checks out.

(c) For $m = 1234$, we compute $H(m) = 9$ and then check, using the public key, whether $H(m) \equiv s^{13} \pmod{55}$.

Doing the usual binary exponentiation, $9^{13} \equiv 14 \pmod{55}$. Since this does not equal $H(m) = 9$, the signature is not Alice's.

(d) For instance, the messages $m = 1$ and $m = 26$ have the same hash value $H(m) = 1$. □

Problem 4.

(a) The movie "Swordfish" features a DoD system using 128 bit RSA, which is broken by one of the actors. What is your reaction to that?

(b) Is SHA-2 considered a secure password hashing algorithm?

(c) What does it mean to salt a password?

(d) In which sense are MD5 and SHA-1 broken? For which purposes must they not be used anymore? For which purposes is it still acceptable to use these hash functions?

(e) Explain why using a hash with, say, 64 output bits is completely inappropriate for digital signatures.

Solution.

(a) Looks like someone confused AES and RSA. While AES exists in a 128 bit version, key sizes for RSA are at least 1024 bit. 128 bit RSA would not provide any security.

https://blog.cryptographyengineering.com/2012/01/30/bad-movie-cryptography-of-week/

(b) No, it is too fast.

(c) It means that we don't compute the hash $H(m)$ of a password m but instead $H(s, m)$ where s is some random data, called the salt. We must then pass on both s and $H(s, m)$.

(d) MD5 and SHA-1 have been demonstrated to not be collision-resistant.

As a consequence, they must not be used for applications like digital signatures, which rely on collision-resistance.

However, MD5 and SHA-1 are still considered preimage-resistant. Hence, it is acceptable (and still widespread practice) to use these hash functions for file integrity checking or, when iterated sufficiently to slow them down, for password storage.

(e) Digital signatures require a collision-resistant hash function. Using a birthday attack, for a hash with just 64 output bits, a collision can be found by computing about 2^{32} hashes. That's easily doable (brute-force becomes difficult around 2^{64} cases). □

Problem 5.

(a) Does Alice have to choose a new y if she sends several messags to Bob using ElGamal? Explain!

(b) Can encryption and/or decryption of RSA be sped up by the Chinese Remainder Theorem?

(c) Let (N, e) be a public RSA key and d the corresponding private key.

It is commonly stated that $ed \equiv 1 \pmod{(p-1)(q-1)}$, where p, q are the prime factors of N. Give a stronger congruence which actually holds.

(d) Give two examples of side-channels that can be exploited in a side-channel attack.

(e) What is a NOBUS backdoor?

Solution.

(a) Yes, she absolutely has to randomly choose a new y every time! Here's why:

If she was using the same y to encrypt messages $m^{(1)}$ and $m^{(2)}$, Alice would be sending the ciphertexts $\left(c_1^{(1)}, c_2^{(1)}\right) = (g^y, g^{xy} m^{(1)})$ and $\left(c_1^{(2)}, c_2^{(2)}\right) = (g^y, g^{xy} m^{(2)})$.

That means, Eve can immediately figure out $c_2^{(1)} / c_2^{(2)} = m^{(1)} / m^{(2)}$ (the divison is a modular inverse and everything is modulo p). That's a combination of the plaintexts, and Eve should never be able to get her hands on such a thing.

(Also, Eve would know right away that Alice is doing the mistake of reusing y because $c_1^{(1)} = c_1^{(2)}$.)

Comment. The situation is just like for the one-time pad (in that case, reusing the key reveals $m^{(1)} \oplus m^{(2)}$).

(b) Decryption can be sped up using the CRT, but encryption cannot. That's because using the CRT requires knowledge of the factorization of $N = pq$. (See Example 171(a).)

(c) $ed \equiv \pmod{\operatorname{lcm}(p-1, q-1)}$ (See Example 171(b).)

(d) Typical examples of side-channels include timing or power consumption.

(e) A NOBUS backdoor ("nobody but us") is a backdoor into a cryptosystem which can only be used by the person who knows a secret (which is infeasible to obtain by anyone else even if they know about the backdoor).

The term is also used to describe vulnerabilities in a cryptosystem which only a powerful agency (like the NSA) has the capabilities to exploit (in which case the agency might be happy to keep these vulnerabilities alive).

https://en.wikipedia.org/wiki/NOBUS □

Problem 6.

(a) A hash function $h(x)$ is called one-way if [ ].

(b) A hash function $h(x)$ is called (strongly) collision-resistant if [ ].

(c) Does using a hash function provide authenticity?

(d) What's the difference between a compression function and a hash function? Which construction allows us to create the latter from the former?

(e) Let E_k be encryption using a block cipher (like AES). Is the compression function $\tilde{H}$ defined by

$$\tilde{H}(x, k) = E_k(x)$$

Armin Straub
straub@southalabama.edu

one-way? If it isn't, suggest a variation which is expected to be collision resistant?

(f) List the main ideas for storing human passwords for authentication.

(g) You need to hash (salted) passwords for storage. Unfortunately, you only have SHA-2 available. What can you do?

(h) Both digital signatures and MACs provide authenticity. What aspect of authenticity do digital signatures provide that MACs don't?

(i) Let H be a cryptographic hash function. What is a simple way to construct a MAC from H?

(j) We have learned about the birthday paradox. What is its implication for hash functions?

Solution.

(a) $h(x)$ is called one-way if, given y, it is computationally infeasible to compute m such that $h(m) = y$. Such a function is also called preimage-resistant.

(b) $h(x)$ is called (strongly) collision-resistant if it is computationally infeasible to find two messages m_1, m_2 such that $h(m_1) = h(m_2)$.

(c) No, everybody can use the same hash function. To provide authenticity, a digital signature or a MAC can be used.

(d) A hash function H is a function, which takes an input x of arbitrary length, and produces an output $H(x)$ of fixed length, say, b bit. On the other hand, a compression function $\tilde{H}$ takes input x of length $b + c$ bits, and produces output $\tilde{H}(x)$ of length b bits.

One popular construction to create hash functions from compression functions is the Merkle–Damgård construction (see Example 187).

(e) No, it is not one-way.

Indeed, given y, we can produce many different (x, k) such that $\tilde{H}(x, k) = y$ or, equivalently, $E_k(x) = y$. Namely, pick any k, and then choose $x = D_k(y)$.

On the other hand, the compression function $\tilde{H}$ defined by

$$\tilde{H}(x, k) = E_k(x) \oplus x$$

is usually expected to be collision-resistant (see Example 189).

(f) Instead of passwords m, the hashes $H(s, m)$ should be stored together with s, a unique (typically random) value called "salt". Moreover, it is important to use a (slow!) hash function H designed for password storage. The usual hash functions like SHA-2 are too fast (thus making brute-force attacks practical).

(g) Iterate many times! (In order to slow down the computation of the hash.) The naive way would be to simply set $h_0 = H(m)$ and $h_{n+1} = H(h_n)$. Then use as hash the value h_N for large N.

In current applications, it is typical to choose N on the order of $100,000$ or higher (depending on how long is reasonable to have your user wait each time she logs in and needs her password hashed for verification).

(h) A MAC does not offer non-repudiation because several parties know the private key. Hence, it cannot be proven to a third party who among those computed the MAC (and, in any case, such a discussion would make it necessary to reveal the private key, which is usually unacceptable).

(i) We can simply produce a MAC $M_k(x)$ (usually referred to as a HMAC) as follows:

$$M_k(x) = H(k, x)$$

This seems to work fine for instance for SHA-3. On the other hand, this does not appear quite secure for certain other common hashes. Instead, it is common to use $M_k(x) = H(k, H(k, x))$ (as well as certain padding).

https://en.wikipedia.org/wiki/Hash-based_message_authentication_code

(j) For collision-resistance, the output size of a hash function needs to have twice the number of bits that would be necessary to prevent a brute-force attack. □

Armin Straub
straub@southalabama.edu