Last time

Classical cryptography ⇒ one-time pad

Block ciphers

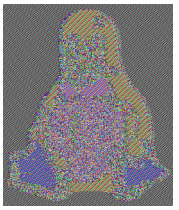
Today: Modes, MACs and hash functions

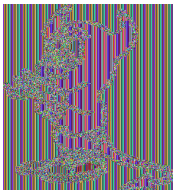
Block cipher modes

Sounds pretty secure, right?

Uhhh...

- passing the same plaintext to a block cipher with the same key will yield the same ciphertext output
- block ciphers alone lacks *semantic security*





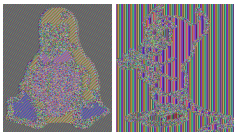
Can you tell which of these is m_0 and which is m_1?

Encrypted images generated with encrypt-image.py

Semantic security (see: Encyclopedia of Cryptography and Security, 2011 Edition, Springer) is defined as the *indistinguishability* of encryptions, i.e., an adversary cannot tell which of two candidate plaintexts has been encrypted to ciphertext.

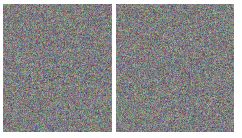
Block cipher modes

Electronic codebook (ECB) mode



- "bare" block cipher
- encrypt each chunk of plaintext directly

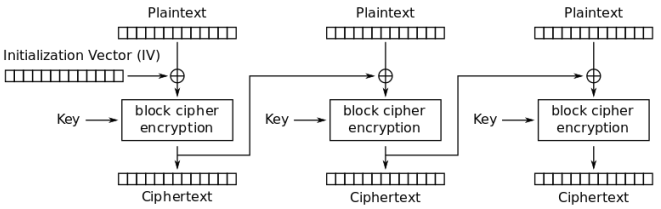
More sophisticated modes



- provide semantic security
- e.g., Cipher Block Chaining (CBC)

Block cipher modes are schemes for handling _____ blocks of plaintext and ciphertext. There are lots of modes (ECB, CBC, CTR, GCM, XTS, ...), each of which can be used with _____. So, to identify a cipher, we need more than just the _____ (e.g., AES): we also need to specify the _____. For example, AES-128-CBC is different from AES-128-GCM.

Cipher Block Chaining



Cipher Block Chaining (CBC) mode encryption

Ciphertext depends on *all* previous blocks

The fact that each block of ciphertext depends on all previous blocks is an example of _____ at work.

Other modes

CTR and GCM modes

- used to make **stream ciphers** out of block ciphers

XTS mode

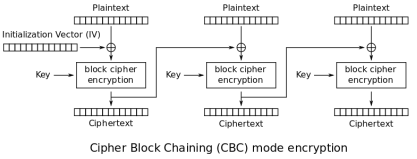
- used for full-disk encryption

... and many others ...

Message Authentication Code

What if we:

1. encrypt in CBC mode and
2. throw away most of the ciphertext?



Cipher Block Chaining (CBC) mode encryption

Message Authentication Code (MAC):

- *cryptographic* checksum that can verify message integrity **even** in the presence of an attacker (vs. checksum like CRC32)

MAC Requirements

1. Arbitrary-length message
2. Small, fixed MAC length
3. Computationally efficent
4. *Collision resistance:*
    - can't generate another message with the same MAC
    - can't generate another message with any valid MAC

Note: the Sealed Authenticator System (SAS) codes on a nuclear-armed submarine probably don't use keyed MACs, but rather purely-random codes that no human eyes have ever seen. Source: Waller, "Practicing for Doomsday", Time Magazine, 4 Mar 2001.

MAC generalization

Newer modes:

Authenticated encryption with associated data (AEAD)

What if we don't want to use a key?

But why wouldn't we want to use a key?

AAA[A]

Category	Question
Authentication	Is something/someone authentic (is it really you)?
Authorization	Are you allowed to do that?
Accounting	Who has used which resources?
Audit	Who did what to what?

Message **authentication vs** ***principal*** **authentication**

Examples of _____ include the authenticated orders in *Crimson Tide* and the payment authorization messages described by the EMV protocol. In both of these cases, there are _____ required besides the _____ itself.

When authenticating _____ instead of _____, we can use messages in which the message itself is the secret, for example...

Passwords

Old and terrible, but...

Dictionary attack

- online
- offline – ???

We'll talk later in the term about protocols that we can use for authentication based on a third party, but at some point, *somebody* has to store a password

A dictionary attack is a brute-force attack: instead of trying every possible key for a cipher, you try every possible password from a dictionary. This is generally cleverer than trying "aaaaaa", "aaaaab", etc., as some passwords are (unfortunately) likelier to be chosen than others. Also, the dictionary may include more than just "dictionary" words!

Threats to authentication

External threats

- password guessing
- MAC-based challenge/response guessing — human-computable?

Internal threats

- password database could be stolen
- ... but so could a secret key for validating MACs!

MAC-based schemes only work when the secret key _____. We can't guarantee that in general-purpose computers.

We'll talk later about public-key schemes that can help with the theft issue, but they don't help with the human-computability problem.

Cryptographic hash functions

Remember hash tables' hash functions?

- variable-length input
- fixed-length output

Cryptographic hash functions

MD4, MD5, SHA-1, RIPEMD-160, Whirlpool, SHA-2 (SHA-224, SHA-256, SHA-384, SHA-512, SHA-512/224, SHA-512/256), SHA-3, BLAKE2/3...

These properties sound like some of the properties of MACs: variable-length input, fixed-length output, computationally efficient and avoiding collisions. However, while regular hash functions try to avoid collisions, they do happen, because the consequences of a collision aren't terribly serious. If we start to see lots of collisions in a hash table, we can always increase the size of the table.

Cryptographic hash functions, however, are something entirely different. A cryptographic hash function should still be fairly efficient to compute (in practice, we can hash millions of MB/s), but efficiency has to be traded off for *much* stronger _____. Once we start sending messages around with cryptographic hashes, we can't recall all of the messages and re-hash them. Instead, we must be very strict about _____ up front.

Cryptographic hash function

Diffusion: small changes ⇒ large effects

All values should be equally likely

Should resist:

Collision attack: find X_1, X_2 s.t. $h(X_1) = h(X_2)$
Preimage attack: given $h(X_1)$, find X_2 s.t. $h(X_1) = h(X_2)$
2nd preimage attack: given X_1, find $X_2 \neq X_1$ s.t. $h(X_1) = h(X_2)$

Collision attack

Finding _____ that hash to the same value. When we get to digital signatures, we'll see that collision attacks can be quite important: if you can generate two messages with different meanings but the same hash, you can cause a lot of trouble! However, such attacks aren't so useful for password security.

Even with the strongest hash function, collisions are _____ _____ due to the birthday paradox. However, "easier" doesn't have to be "easy": if the hash output is large, you can still have a lot of work to do! $\sqrt{2^n}$ can still be a large number if n is big enough...

Preimage attack

Finding an input that hashes to the same value as a given hash. This could be the same input that was originally used to generate the hash or a different one.

Second preimage attack

Finding a _____ input that will hash to the same value as a given input. This is like a collision attack, but much harder: instead of generating lots of messages and finding two that hash to the same value, you have to find one that hashes to the same value _____.

Password hashing

What does this have to do with passwords?

Resisting *offline* dictionary attacks*

Rainbows† and salt

Iterative password hashing (KDFs)

---

* see, e.g., John the Ripper

† Oeschslin, "Making a Faster Cryptanalytic Time-Memory Trade-Off", CRYPTO 2003: Advances in Cryptology - CRYPTO 2003, 2003. DOI: 10.1007/978-3-540-45146-4_36.

We don't need *any* cryptography to resist an online dictionary attack. Protecting password databases is, instead, all about resisting _____, where an adversary has gained access to a password database and they want to get passwords from it. Without any cryptography, they can simply do a database lookup. With cryptography, however, we can make things much harder for them.

As a (very bad!) alternative to password hashing, check out this analysis of a major password database breach at Adobe.

Tools like GPUs are really good at parallel computation. Attackers can use them to try lots and lots of passwords concurrently to see if they can find the correct one (a bit like the Bombes in Bletchley Park!). _____ (KDFs) make life harder for an attacker by forcing computation to be _____. There is a cost for the user, too, but it's insignificant compared to the benefit of not having your password cracked when a business suffers a data breach!

What makes a good password?

(we'll answer this next time)

MAC generalization

~~What if we don't want to use a key?~~

What if we don't use a block cipher?

HMAC: hash-based message authentication code*

$$h\left((k \oplus p_o)||h((k \oplus p_i)||text)\right)$$

* Bellare, Canetti and Krawczyk, "Keying Hash Functions for Message Authentication", CRYPTO 1996, 1996. Standardized by NIST (FIPS 198-1) and the IETF (RFC 2104).

An HMAC uses a hash function *with* a key. This provides the same security properties as a block-cipher–based MAC, just with a different underlying cryptographic algorithm. HMACs are pretty popular in circumstances where you'd be doing a bunch of hashing anyway (e.g., Transport Layer Security cipher suites, which we'll talk about later).