CS 240: Cryptographic Hash Functions Transcript

[00:00:00] The first core security concept we want to cover is cryptographic hash functions.

Start visual description. The professor demonstrates the concept of cryptographic hash functions, explaining their importance in secure computing. End visual description.

[00:00:04] Now, hopefully from your uh CS 235 or your data structures course that you've previously taken, you have some notion of what a hash function is.

[00:00:11] I take a piece of data, I run it through a function which basically gives me a, a number um that represents the, the data value that I passed through the hash function and cryptographic.

[00:00:23] Typically, when people say cryptographic, they're just trying to say it's really secure and it's not easy to crack or to hack or to break.

[00:00:29] And so cryptographic hash functions are really at the foundation of a lot of the things that we do in secure computing.

[00:00:37] So the idea with a cryptographic hash function and these are sometimes called one way hash functions.

[00:00:44] But the idea is that if I have a, a body of data and the data can vary in size, it could be relatively small like one or two K of bytes or it could be a gigabyte of data uh really any size of data.

[00:00:58] And what, what you can do with a cryptographic hash function is you can take your body of data, think of the data really is just a big array of bytes and I pass it through the cryptographic hash function which is represented by this yellow rectangle.

Start visual description. The professor shows how a cryptographic hash function processes a body of data, using the example of the word "fox" to illustrate how the function outputs a fixed-size digest. End visual description.

[00:01:12] So I take my data which in this case is just the word fox and I run it through the cryptographic hash function.

[00:01:18] And the output of the cryptographic hash function is a fixed size digest of the original data that I passed through the function.

[00:01:27] Now, when they say digest, they really just mean it's a summary of the data or maybe you could think of it as a fingerprint.

[00:01:35] So the, the digest that's created by them, the hash function is in its fixed in length.

[00:01:42] For example, it might be 100 and 28 bits long or it might be 100 and 60 bits long or something like that.

[00:01:48] So no matter how big the data that you sent into the hash function is the output of the function is fixed in size 100 and 60 bits, for example.

[00:01:58] And so think of this value as sort of a random number that represents in a small way, it summarizes the contents of the original data.

[00:02:08] So it's kind of like a fingerprint and that this, this digest is sort of it sort of uniquely identifies the data, uh the original data that was passed into the function.

[00:02:20] Now, um not only is the output of the hash function fixed in size.

Start visual description. The professor explains the one-way nature of cryptographic hash functions, emphasizing that it is computationally infeasible to recreate the original data from the digest. End visual description.

[00:02:25] But the function is, is what's called one way. So, these are one-way functions.

[00:02:29] And what that means is if I give you the digest of the data, there's, there's no way that you can reasonably recreate the original data from the digest.

[00:02:41] So the, the function goes in only one way it goes from the data to the digest.

[00:02:46] But if I give you a digest, there's no way you can get back to the data, at least it's not computationally feasible given, given current technology.

[00:02:54] And that's very important to make this secure that and the security of these hash functions is really based in this idea that you can't take the, the digest and, and somehow recreate or reverse engineer the original data.

[00:03:08] Another important property of these hash functions is that they're deterministic.

[00:03:12] Meaning if I take the same data and pass it through the same hash function, I should always get the same message digest.

[00:03:19] So there's no randomness really, it is deterministic.

[00:03:22] Even though the digest looks like a random number, it's really not.

[00:03:25] It's, it's truly based algorithmically on the original data.

[00:03:29] And you can see here that we passed the word fox through the hash function twice.

[00:03:33] And you can see that the digests are, are equal. And so that's, that's essential.

[00:03:41] Another thing that's important with these functions is that they're statistically random.

[00:03:47] I mean, they really do look random meaning if I have some data.

[00:03:50] Like in this case, I have the red fox jumps over the blue dog pass that through the hash function. I get a particular digest in, in the next example, you can see that we've modified just one letter in that, in that data.

Start visual description. The professor demonstrates the deterministic property of hash functions by passing the same data through the hash function twice and showing that the digests are equal. End visual description.

[00:04:02] So it says the red fox jumps ou er so it's not over, it's our the blue dog and just having made that one-character change, the digest that the data creates is totally different.

[00:04:14] And so, so that's an important property as well.

[00:04:18] So um smart people invent these cryptographically secure hash functions and then we, we base our a lot of our security on them.

[00:04:30] So, so what are the commonly used cryptographic hash algorithms? Well, historically, uh some of the earlier algorithms were MD five. So, MD stands for message digest.

[00:04:42] Uh MD five was an algorithm invented at MIT by uh Ron Rivest and it produced 128-bit digests.

Start visual description. The professor discusses commonly used cryptographic hash algorithms, mentioning MD5, SHA-1, and the more secure SHA-2 family of algorithms. End visual description.

[00:04:50] And there was also one called sha another algorithm called sha one sh A that stands for secure hash algorithm, and it produced 160 bit digests and it was designed by the National Security Agency and both of these algorithms have been um cracked. So, there are known ways that you can compromise these algorithms. So, they're not used in highly secure applications anymore, but you'll

still see less secure applications that make use of these algorithms, or you'll see older, older um software applications that use these algorithms.

[00:05:25] So it's helpful to know what they are.

[00:05:27] So the algorithms that are used more modernly would be um Sha two, which is the currently the most commonly used C cryptographic C function.

[00:05:41] And so this is kind of the descendant of MD five in Sha one.

[00:05:47] So Sha two is actually a family of algorithms, meaning that you can use different versions of Shaw based on how big you want the message digest to be. And so Sha 256 is, is very commonly used in, in the industry and it's, it's very secure.

[00:06:06] And what that means is that the message digest it creates is 256 bits.

[00:06:11] So Sha 256 is quite secure.

[00:06:14] But if you need even more security than that, you can use Sha 384 or Sha 512.

[00:06:19] So there's a lot of options and it just depends on how much security you need and how long you want to wait for the algorithm to complete.

[00:06:26] And um there's even another family of algorithms called Sha three.

[00:06:31] So people are always working on inventing new algorithms that are better than the current ones.

[00:06:35] And so um there's no end to the uh algorithms that are out there and that people are working on.

[00:06:42] But currently it's the sha two family of algorithms is uh perhaps the most commonly used.

[00:06:51] So what I want to do now is show you a code example.

[00:06:55] Uh as we go through these topics, I want to show you some code examples just to show you how easily you can actually incorporate these, these security concepts into your software.

[00:07:04] And so we're going to switch over now to IntelliJ and we're going to look at um one of the code examples that comes with the security lecture.

[00:07:12] So all these code examples are on, and you can find them there.

Start visual description. The professor switches to IntelliJ to show a code example of how to incorporate SHA-256 hash functions into software, demonstrating the process of hashing data and printing out the hash. End visual description.

[00:07:18] So the first one I want to look at here is called crypto hash function demo.

[00:07:23] And so this is just a little program that takes some data and runs it through the, the sha 256 hash algorithm.

[00:07:31] And so what we have first here is we have just an array of strings.

[00:07:35] So these are the same strings that I showed you in that previous slide.

[00:07:39] And what we're going to do is we're going to take each of these strings and, and, and run them through the, the Shaw 256 algorithm and then we'll see what their message digests look like.

[00:07:48] And so we just have a simple array of strings here and then we have a little four loop that just iterates over those strings. And it, and each one of those strings is then hashed using, using Shaw 256.

[00:08:00] So let's see how that works. The first thing we do for a string is we have to convert it to an array of bytes because all these algorithms work on bytes, they don't really work on characters per se.

[00:08:10] So we have to take our strings and convert them to bytes.

[00:08:13] And so the first thing we do is we, we take the, the string and we call, get bytes on it.

[00:08:18] And that gives us back an array of bytes that represent that string.

[00:08:21] And then we, we go ahead and hash the data by calling the hash data method.

[00:08:28] And you can see here how to actually do the hashing.

[00:08:30] So in Java, they have a class called message digest.

[00:08:33] And you can see if you look at the imports up here that Java has a, a package called java dot security.

[00:08:39] And so there's all manner of cool security stuff in, in that package.

[00:08:44] And so it's, it's uh it's interesting to, to read and learn about the classes that are in that package.

[00:08:49] But one of those classes is called message digest.

[00:08:54] And so uh message digest has a, a static method called get instance where you pass in the name of the algorithm you would like an instance of.

[00:09:01] So we pass in Shaw 256, and we get back a message digest object which represents the message digest algorithm.

[00:09:09] And then once you've got a, a reference to the algorithm, you just call the digest method on it and pass the data to it that you want to, to hash and then it gives you back an array of bytes which represents the message digest for, for that data.

[00:09:23] And so then in, in this case, after we, we hash the data, we just go ahead and print out the hash.

[00:09:29] So if we run it, let's go ahead and run it and see what the output looks like.

[00:09:35] And so you can see that uh for the word Fox, we get this thing as our message digest looks very random.

[00:09:42] There's really no way you could take that and, and reverse engineer it and get back to Fox.

[00:09:46] You can see that uh Fox produces the same output both times, et cetera.

[00:09:50] And so that just gives you a sense of what it looks like.

[00:09:52] But really what we're dealing here with is just a ray of bytes.

[00:09:56] So it's not very hard to incorporate this kind of thing into your applications.