**First lecture:**

The year is 1995. There is an internet. It’s been around since the 1970’s. People are connected to the internet using modems. Sometimes they have ISDN. How do they use the internet? They have browser called Netscape Navigator (figure 1), and there are browser wars. Now there's a new browser called Internet Explorer 3 by Microsoft. How do people use cryptography in the year 1995. Well cryptography was used to be a military technology and it became a surveillance technology around 1980 1984 with a crypto conference the first conference on cryptography for academics and people are encrypting **(CRYPTO**, the **International Cryptology Conference**, is one of the largest [academic conferences](https://en.wikipedia.org/wiki/Academic_conference" \o "Academic conference)in [cryptography](https://en.wikipedia.org/wiki/Cryptography" \o "Cryptography) and [cryptanalysis](https://en.wikipedia.org/wiki/Cryptanalysis" \o "Cryptanalysis). It is organized by the [International Association for Cryptologic Research](https://en.wikipedia.org/wiki/International_Association_for_Cryptologic_Research" \o "International Association for Cryptologic Research) (IACR), and it is held yearly in August in [Santa Barbara](https://en.wikipedia.org/wiki/Santa_Barbara,_California" \o "Santa Barbara, California), [California](https://en.wikipedia.org/wiki/California" \o "California) at the [University of California, Santa Barbara](https://en.wikipedia.org/wiki/University_of_California,_Santa_Barbara" \o "University of California, Santa Barbara)). Even the browser has encryption. However, encryption is considered ammunition by the U.S. government. And so if you download the browser from the U.S. you get it with very very weak encryption - 40 bits which is very weak. You have to download the encryption part of it from Finland because Finland was legal. In 1996, Bill Clinton made it legal to export our encryption.

 Icon used on [Netscape Navigator](https://en.wikipedia.org/wiki/Netscape_Navigator" \o "wikipedia:Netscape Navigator) browsers from 1994 to 2007.

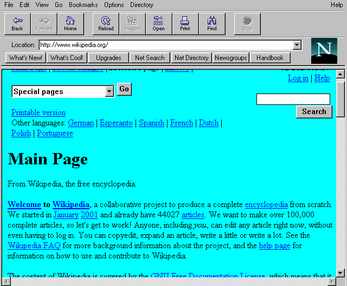


Figure 1 - Netscape Navigator 1.22

(Slide 2) This is the front page of The New York Times of December 11 1995 ( figure 2). Now you can see here Shimon Peres. We can go to the Web site and you can see why it's Shimon Peres and what was he doing in New York (Nearly 15,000 people crowded into Madison Square Garden to honor the memory of Yitzhak Rabin, braving a long, bone-chilling wait outside to accord the slain Israeli Prime Minister and his quest for peace the kind of unanimous mass tribute from American Jewry he never enjoyed while alive.

Several thousand were turned away at the arena's doors and missed sorrowful speeches by Prime Minister Shimon Peres of Israel, Vice President Al Gore and the widowed Leah Rabin. This is Hong Kong and this is the headline on the front page of the New York Times in December 11 1995. The secure digital files just got a little less secure. What does that mean? So there was a researcher called Paul Kocher, his age was 22. He lives in San Francisco and he just released a paper called Timing attacks on implementations of Diffie-Helman, RSA, DSS and other systems [1]. And he basically said that he can attack all of the modern cryptography. He published this paper before he published it online. He wrote about it to a mailing list. In these days there was a mailing list called Cypherpunks which is a mailing list which includes all sorts of people who were entering the cyber neuro, there mathematicians, photographers but also artists and anarchists and so on (A **cypherpunk** is any activist advocating widespread use of strong [cryptography](https://en.wikipedia.org/wiki/Cryptography" \o "Cryptography) and [privacy-enhancing technologies](https://en.wikipedia.org/wiki/Privacy-enhancing_technologies" \o "Privacy-enhancing technologies) as a route to social and political change. Originally communicating through the Cypherpunks [electronic mailing list](https://en.wikipedia.org/wiki/Electronic_mailing_list" \o "Electronic mailing list), informal groups aimed to achieve privacy and security through proactive use of cryptography.).



Figure 2 - December 11, 1995, Page 1. The New York Times Archives

(Slide 3) here is this posting from 1995 (figure 3). I think it's fascinating to see people responding to this. And what do you think they responded? they say of course it's bullshit and it's not the real attack and obviously we are secure. They say: Why would somebody connect a secure computer to the Internet. It's ridiculous. The Internet is not secure. But this is a temporal side channel. It's the most simple to explain side channel. So I will begin my discussion of such channels with this. We will start with a very simple example which we will also have to solve at home. And then we move onto cryptography. So by the way the company which paul kaucher established on Cryptography Research was later sold to Rambus for 400 million dollars. So he was doing pretty nice.   
Paul Kocher did another timing attack a couple of months ago called Spectre and meltdown (Vulnerabilities in modern computers leak passwords and sensitive data). He was one of the co-authors and obviously I have to teach it in this course. I will show you later in this course how this timing attack connects back to Meltdown and Spectre.

So the definition of a temporal attack is that you can get confidential information based on the time it takes to complete an algorithm (from the beginning to the end of the algorithm). So he proposed the paper in 1995, and people said OK you know everybody writes papers and in 1997 there was a cryptography conference called crypto and then the end of crypto conference, there was something called the rump session which is like a you have you can get very short in formal talks for like three to five minutes, and somebody went on stage and demonstrated a timing attack live. So this was the first time it was demonstrated live in 1997 and later in 1998 somebody wrote a paper describing: here is how you do from A to Z - a Practical timing attack starting from the measurements and ending with the key. Every time you add a new feature to your system: new accelerator, new speed up, a new trick etc. you are exposed again to timing attacks. SSL was broken more than once using timing attacks - 2003 2007 etc.

So a timing attack measures the time it takes from the beginning of the algorithm to the end of the algorithm. Cache attacks, which we will go into detail later in the course, are special case of timing attacks. They measure how long it takes to execute a particular part of the algorithm. Which of these attacks is more powerful? Student said cache attack. Why? Because we can we can zero down on a specific part of the algorithm. But on the other hand what will the system engineer say: You cannot do cash attacks on my system because I am protected using methods A B C. timing attacks you can do. And one of the things I'll talk about cache attacks is very scary how practical it is to run them. So I'm going to show you now a timing attack on passwords just to understand the language and the general thought. And then I'll talk about RSA. So there is going to be algebra today.

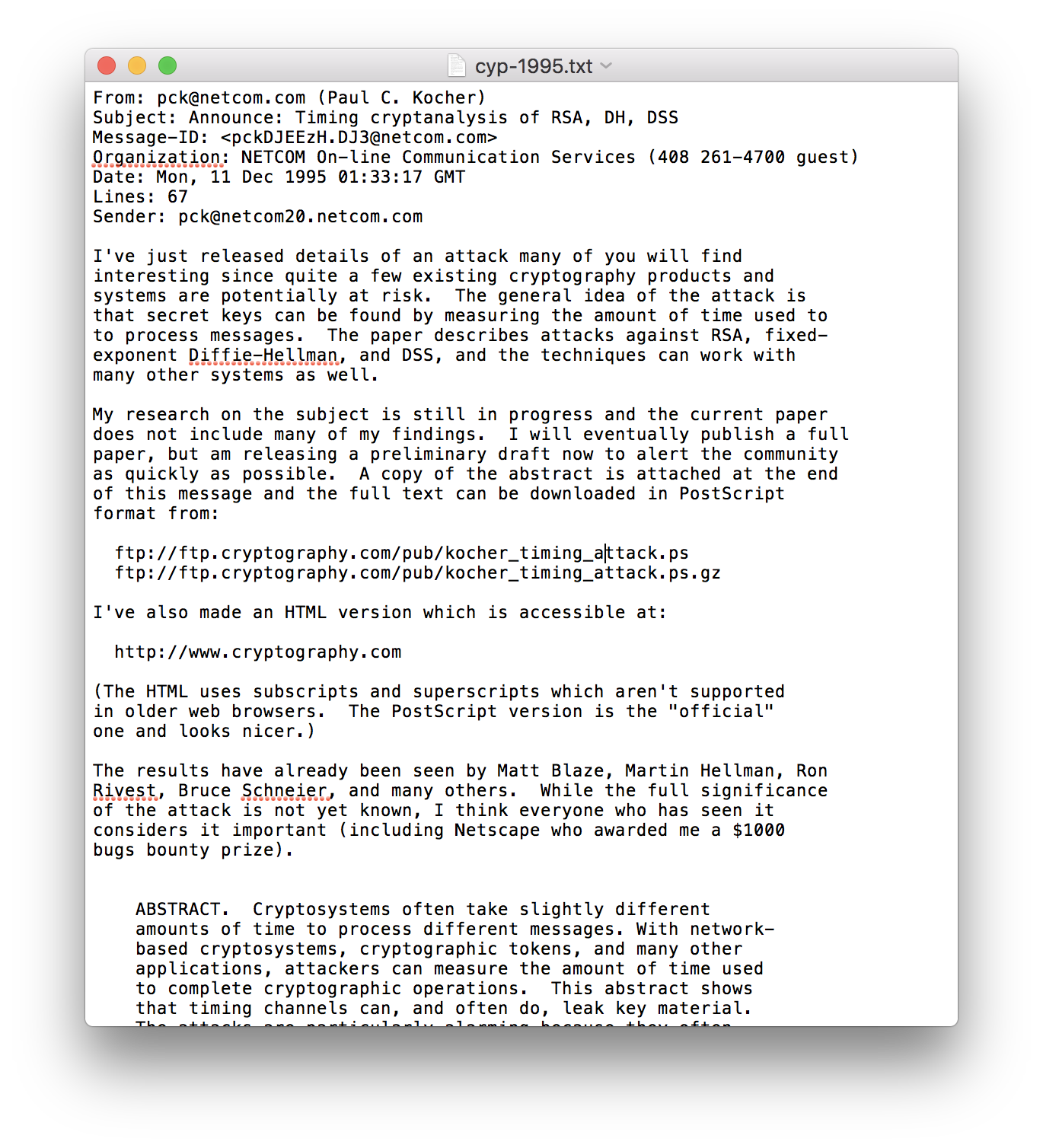


Figure 3 – Posting from Paul Kocher in the cypherpunk, 1995

(Slide 7) Here is the threat model (figure 4) [2]. So what do we see here. Here is a system, and it has an implementation in it. And the implementation has an asset. What is the asset? It is the secret. And what are the capabilities of the attacker? the attacker can send a question to the implementation and get an answer from the implementation and the attacker is also able to measure the time difference between the request and the response. What is the objective of the attacker? in this case the objective is to recover the secret. In general case if I have a password access to a system and I am running a timing attack on the system maybe my objective is not to recover the password, it is just to get into the system. But in this case I want to send a question get an answer and measure the time difference between the two. First of all is there a scenario where you cannot find time – you cannot do timing attack. A student says: This happens where the time is not dependent on the computation. But let's assume the submit implementation is not particularly secure and I still want to talk about the system. So if I'm talking about an algorithm, a theoretical algorithm, then I cannot do this. This has to be a device. And I can also talk about something called a trusted third party, like an idealized implementation in that anybody do an exam where the the professor did not go with the class. Did anybody have these exams. So usually the teacher goes between the seats and asks questions. But you know that the professor also had side channels: He can smile, He can point, etc. So sometimes there are classes where the professor did not go into the class. He sits outside, the T.A. listen to the questions from the students and then the T.A. go outside and talk. So the professor is outside getting drunk and the T.A.s are walking in and out. So this is a trusted third party. So let's say what I write on my note is the password five. And I give it to the T.A. and the T.A goes outside and he comes back in. And he said OK here's a paper from the professor. It says no. Maybe the T.A. didn't even look at the note. So in this case I could not do a tiny measurement because let's say the rules are that once every 15 minutes the T.A. goes out and after 15 minutes he comes back. So then the idea that idea like system a trusted third party does not.

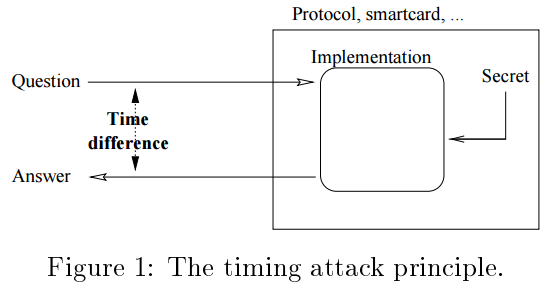


Figure 4 – The timing attack principle

So let's talk about a situation where this is possible. There are three situations in which it's reasonable to run a timing attack. So the first situation is when we actually have the device in our possession. So we're talking about capability of the attacker. So why should we have the device in our possession? Why should there be such a situation? Smartcard!   
Device under test (figure 5) by the way is the politically correct term to refer to a device under attack. Why? Because you're not attacking it you're just testing so why not. What are other cases? So maybe you are using a software library which has a secret and you can use it and you can not decompile it for some reason, because that is not the implementation that I get. But you can obviously have it in your possession and there's a situation where you have the device under the test in your possession.

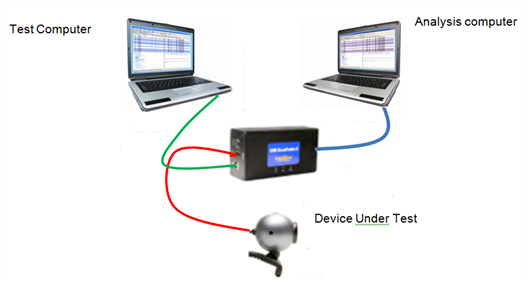


Figure 5 – Example of DUT - Connecting One Device Under Test

Example of one of the students: the phone has what's called the subscriber identity module SIM card inside. And of course you are given this SIM card then if you want to get the secret case of see which is inside the SIM card you can also do things. You can generally do bad things to the phone network. So sometimes you have a device under your position. Another situation is that you are running on the same hardware as the device. What does that mean? Let’s explain with the concept of virtual machine. Let's say I'm running I'm running a store. I want to run the web store and I write down all sorts of PHP stuff, very insecure, with shopping carts and a recommended item and animations. And then I want to get credit card details from the user. But not anybody can get credit card details from the user to be able to process credit cards. You have to pass this very difficult certification process called PCI DSS( **Payment Card Industry Data Security Standard)**. You have what is called Credit Card clearance. This is a computer which can get a list of items and the price of each item and it will display a UI with a payment confirmation. So what you do is you have a system which is running all the non-credit card stuff of your shop and another machine which is doing the credit card processing. And to save money what you can do is you put the VM of the credit card and the VM of the shop on the same hardware. So now if you want to attack this system you're running on the same machine through consent challenges you can you can maybe call an API and see how long it takes for it to come back. And this example of separation using VM. You can also think about user separation or act separation or process separation. For example, maybe there is a service on my computer which unlocks files and I can call it I can't see what it's doing but that could say access granted the access denied. For example, if I can measure this thing. And the third situation which is the most difficult to sell from the systems engineering point is if I can access the device over the Internet. So there is a service running somewhere and I can send queries to the service. The service will return and I can measure how long it takes. Why is this the most unconvincing situation? The main part of the delay here is going to be the communication over the Internet, could be milliseconds going back and forth and maybe like microseconds of distance. So why should it be possible to do timing attacks over the net? So it was proven that you could do timing attacks over the net. Why can you do it? Because of statistics. Everybody ignores statistics let's just say we'll go a little about statistic. You can actually measure timing over the Internet. So another thing we can assume is that the source code of this implementation is known to the attacker. We assume that the only thing not known is the secret key. The objective of the attacker to discover the password. You can also say get entry into the system. And what could the attacker do? The attacker can send unlimited queries and measure their time. Unlimited queries are not so trivial, If you think about it. Some devices can disable themselves after 3 or 4 or 5 periods. This can be a defense (i.e. **[ATM Card Blocked After Certain Numbers Of Wrong PIN](https://www.isrgrajan.com/atm-card-blocked-after-certain-numbers-of-wrong-pin-attempts.html)**[).](https://www.isrgrajan.com/atm-card-blocked-after-certain-numbers-of-wrong-pin-attempts.html)

(Slide 8) Now I'm going to show you a source code of password checking function. I wrote it in JavaScript (figure 6).

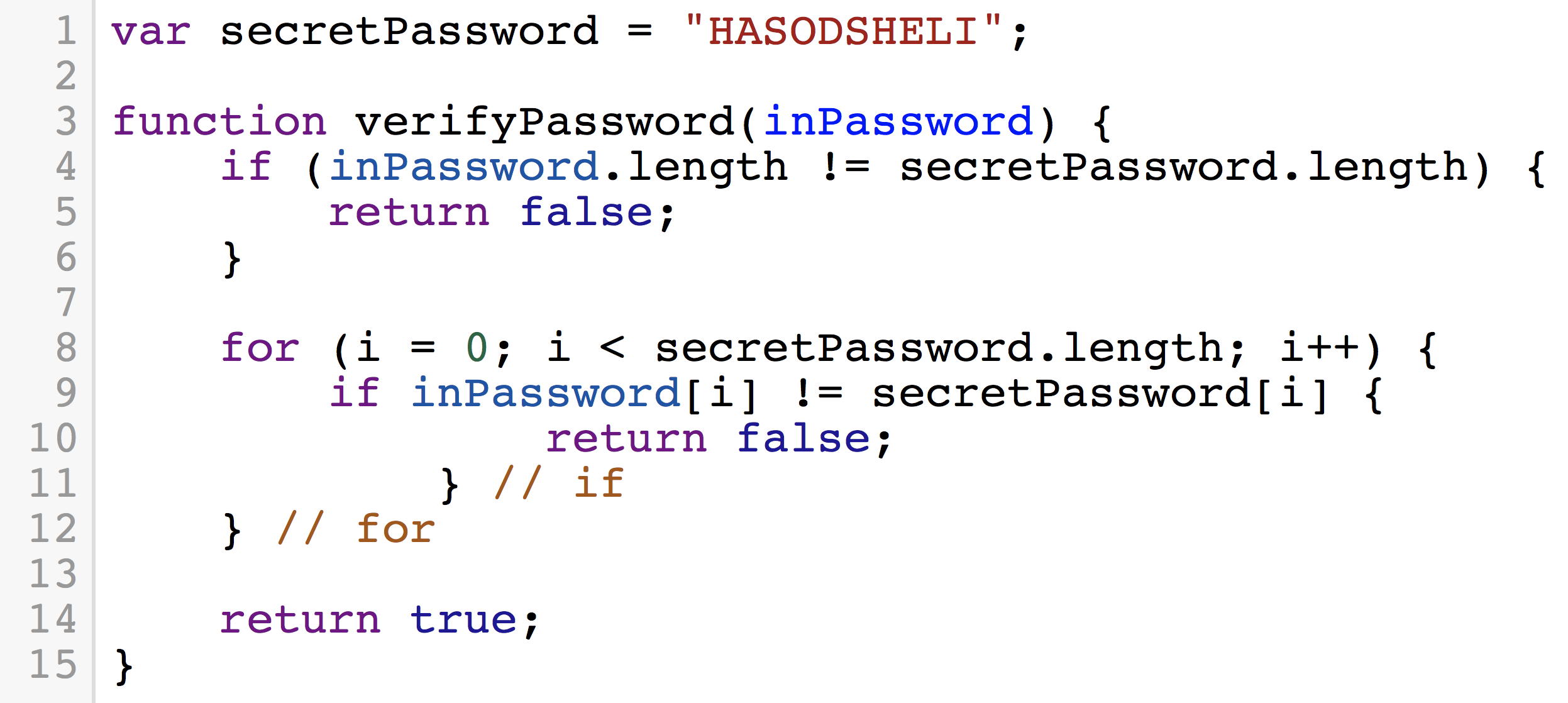


Figure 6 - Password Checker Implementation

There are the secret passwords and there is a function called verify password which gets us input the input password and returns true if this is the right password and false if this is the wrong password. So let's see how this function works. So in line 4 you check if the length of the password which is what is input is identical to the length of the password you’re receiving. Why would I want to do this? What am I protecting against? Buffer overflow. Line 8: I'm going to iterate over the letters in the password and I know it's the same length both of them because I wouldn't have gotten into this line otherwise. Line number nine. I compare the characters of the password in the input one by one. And if there is a different letter one time, even one time, then I exit the function with false and if I get to line 14 what does it mean that all of the letters were identical. It means it's the right password. If I'm running in a situation which timing attacks are not possible, let's say I'm running a trusted third party mode. Or I am incapable of measuring time. So let's assume that that that this system is secure against timing attacks. We can do a brute force attack( try every possible combination of valid characters in order to discover the password). So let's assume the password is no more than 60 characters. How long will it take me to recover the password? how many attempts? 26 to the power of 16. Let's understand why is this case. To get the first character I need 26. The second character 26. the third and so on until the character number 16. And I have to do all of the guesses. I have to get them all of them right at the same time. So 26 to the 16 (. It is a lot a guesses. Using log arithmetic we get it is about 2 to the 75 which is a lot. So now let us attack this using the timing attack. So how to attack this. Line 4 is the first code that runs on this program. And what happens if the length of the password is not equal? it returns false. But it always returns. Why is it so special? it turns out faster. So the password is less than 16. We hold an array of size 16. Each one of these values in the array is the time, each one refers to time in nanoseconds. So now what do I do with this array? The largest number is not necessarily our answer. There is noise. We are doing implementation attacks- we are in the physical world so we might be getting inaccuracies. We can repeat the procedure 100 times. We get a matrix of 16 by 100. We can calculate the average. So how can I be sure that I measured enough times? I am trying to say that one of these rows is sampling a different random variable than all the other rows. We assume (we have hypothesis) that one of the 16 rows is from different distribution. We can say that with a certain probability we are sure that this row is different than the other rows. Why am I sure? because the average is different and also the variance is very large, so there is low probability we are wrong. I am talking about statistics because now we are talking to the systems engineers and say: look! Your system is vulnerable to timing attack. And they say, so what? You have to do million measurements and this system can do only 1000 measurements. And then you say: no! look, if you do 100 measurements then you already have significance. You need to convince like a scholar that this attack is possible. Now after discovering the length is 5 what is the next step? We get to line 8. We will save an array of size 26, where each cell holds time. Our inputs will be AAAAA, BAAAA, CAAAAA… One cell value will be larger because it will repeat line 9 and thus take longer time. We discovered let’s say the first letter is C. Now we have another data structure of size 26. The inputs will be CAAAA, CBAAA, CCAAA…. We are going over the second character. We continue until we get the entire password.

Full Example:

When checking strings for equality we need to check that every character matches. Most programming languages will take a short-cut and return False - or Not Equal - as soon as they find a single character that doesn't match. For example:

str1 = "1111111111111111"

str2 = "1101111111111111"

We know that they are not equal as soon as you hit that 0, why would you keep checking?

This is super exploitable. Imagine that we are trying to crack a password, knowing that the server will do a lazy string comparison somewhere inside, we might get the following timing results:

Tried "aaaaaaaa", it took 0.2 ms

Tried "bbbbbbbb", it took 0.2 ms

Tried "cccccccc", it took 0.4 ms

Now we know that the first letter is 'c' because it took longer. Now I can do the same trick with "caaaaaaa", "cbbbbbbb", etc. Cracking one letter at a time is HUGELY faster than cracking the whole password at a time

How many queries (function calls) did it take to get the password? If the password is N, in the previous algorithm, the brute force, it took

O(constant \* 2^N). In our second attempt it takes O(N). We calculate ln(26\*16)/ln(2) and we get 8.7 (. In total we get 2^8 instead of 2^75. This attack is very impressive. Even if we repeat it 100 times it is not a big deal. We reduced the time it takes the attacker from exponential time to linear time. Why is it so problematic? Let’s say we have a situation where I’m working in my university and I’m reading a lot of papers, and I find out the attackers are able to crack systems with 50 characters in the password. We ask to make are system twice as secure. If the attacks are exponential times, we have to add 1 bit to the length of the password to make it twice hard to attack. If the attacks are in linear times, we have to double the password length. It is problematic since it is very expensive due to the computations involved.

Now let’s talk about defenses. First we talk about two general approaches to counter measurements: mitigation and prevention. Can anybody suggest mitigation counter measurement? We can add a line between line 3 and 4 a sleep (delay) of random number of Nano seconds. What is really nice about mitigation is that we don’t really have to understand the code to implement it. We just have to add a random number generation at the beginning. The problem with mitigation is that the code will get slower. The main problem is that we can add more repetitions and we will solve the problem as attackers. Sometimes mitigation is enough, but it only moves the target a little away from the attacker.

The second type of counter measurement is prevention. In prevention we want to make the system completely resistant to timing attacks.

**Second lecture:**

(Slide 9) Of course prevention is the better one. We make every execution to run the same length and then the attacker has nothing to do. I will ask you for suggestions of how to do it and then I will show you an implementation I wrote which I think is not 100 percent secure. Then we’ll see an implementation I think is pretty good, and then we’ll talk about homework. One student suggests a function that will do garbage steps. That is an option. So I’m going to show you now an implementation which I think is a little bit more secure (figure 7).

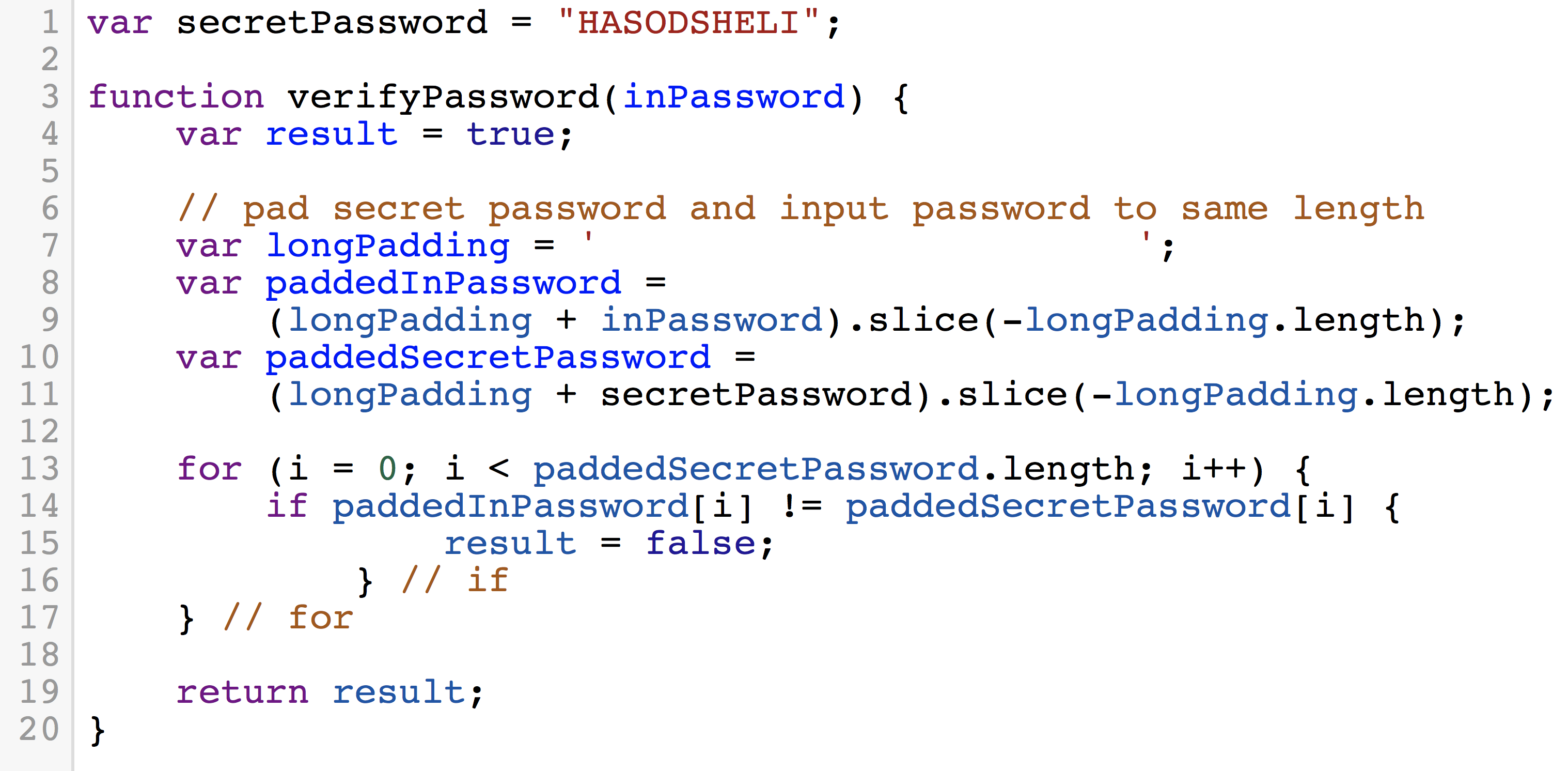


Figure 7 - Password Checker Countermeasure I using same length of secret password and inpt password

So, I have a variable called result in line 4. What I do between lines 6 and 11 is that I make sure that both secret password and input password are the same length. At the end of this, both of them will always be the same length. Which means that I am always going to do lines 13-17 the same number of times regardless of the number of characters in the password (The amount of comparisons can teach you about the length of the password). So we initialize result to true. If the password character is incorrect then result is equal to false. Then we return result. Let’s analyze this code. First of all, we don’t return in the middle of the function. Does anybody see any vulnerabilities in this function? How many times am I accessing the variable result? Every time a character is false. The running time of this function is a function on how many wrong characters I have in my password. This is a more vulnerable implementation than the previous one! Because now I can send n queries and find the password(linear), because every time I send a query my output is not only correct/incorrect but also how many wrong characters did I have. This is even worse than the previous one. The problem is that result variable is written only when the password is incorrect.

How can we fix this? We can add after the if: else {result & = true; } The problem with this is that &= takes longer to run than =. So we can add instead: else { shekerKolsheu = false; }. The problem now is that result=false may take different time than shekerKolsheu=false due to cache. One reason which we will discuss later in the course is called speculative execution(an [optimization](https://en.wikipedia.org/wiki/Optimization_(computer_science)" \o "Optimization (computer science)) technique where a [computer system](https://en.wikipedia.org/wiki/Computer_system" \o "Computer system) performs some task that may not be needed. Work is done before it is known whether it is actually needed, so as to prevent a delay that would have to be incurred by doing the work after it is known that it is needed. If it turns out the work was not needed after all, most changes made by the work are reverted and the results are ignored.). It means that the cpu is trying to guess what the program is going to do next, and execute it ahead of time. So the cpu is looking at this code and is trying to see if it going to the if or the else. When I am going to this line usually my guess is all incorrect, so it says, I go into the if, and then I do result=false, then again go into the if, and result=false, the third time it won’t check, it will write result=false, and then we got a slightly faster execution. If it turns out we had to go into the else, the cpu panics and discards the calculations of getting ready to set result into false, and then it had to write shekerKolsheu=false, which will take much longer because it was all prepared to do result=false. It can do a lot of troubles with our issue. In this case you can do a temporal side channel attack based on the fact that the cpu was not guessing it is going to do the uncommon line.

The right way to store passwords is using hashes (storing the hash of the passwords). What is a hash? A hash is a cryptographic function which has property called the avalanche property – If I flip one bit in the input, half of the bits flip in the output. It means that if I am very close to the password, it doesn’t really matter. I don’t get closer to the target. How we are going to use hashes to do a secure password checking function (figure 8)?

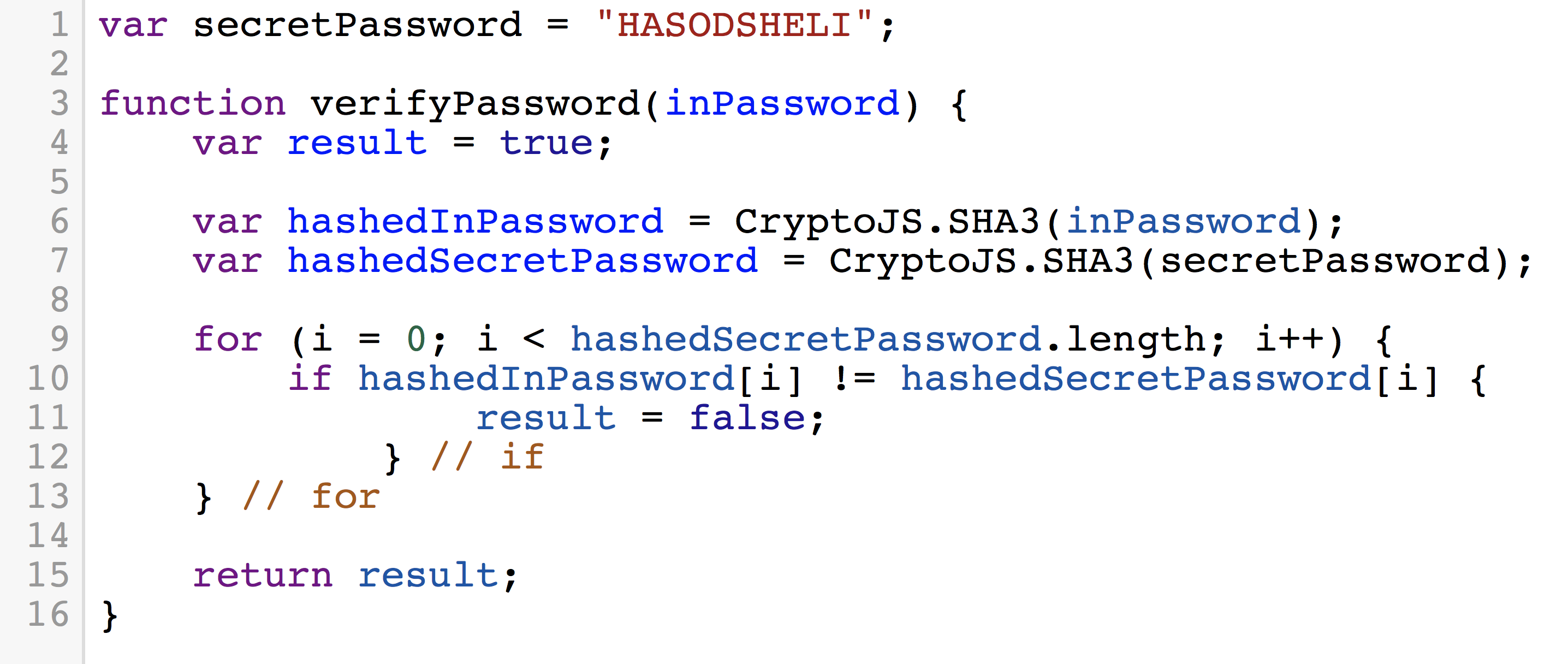


Figure 8 - Password Checker Countermeasure II using hash function

(Slide 10) So result=true. I am going to hash my input password and the secret password. Then I will compare the characters of the hashes. Does it matter to me if the hash is close to the hash of the password? It doesn’t matter because I don’t know how to change the input, the hash is hard to invert. There is still a leak in this function. Line number 7. This line takes longer to run if the password is long. I can’t determine the password but maybe I can determine the length of the password. How to fix this? I can calculate the hash ahead of time. Generally, Linux stores only the hash of the passwords. So now that we know how to do timing attacks and counter measures it’s time for you to hear about homework assignment number 1. Here is a website. This website is accessible for anyone in the world. Let’s go to this website. It is giving user and password. The answer is zero. That’s not the password. Let’s try “yerakot”. Again 0. Your assignment is to find the password. The code that is running is the one in slide 8 (figure 7). The user is going to be your ID number. I am going to show you how to access this using the command line. Running command: curl “aoi.ise.bgu.ac.il/?user=cyber&password=bgu”

Now I will show you a trick to measure time. I can measure time using curl. I am going to do :

curl –s –w “\n%{time\_total} sec\n” “aoi.ise.bgu.ac.il/?user=cyber&password=bgu”

you will submit a function that will calculate the password. The faster your function runs the higher grade you’ll get. There will be two milestones.

Efficient Implementation of RSA

Now let’s talk about some algebra. But I want to talk with you about RSA [3], because the next thing we are going to attack with timing attacks is RSA. So, RSA cryptosystem lives in something called a multiplicative group. The group that it lives in is Z\*n. We take two prime numbers, multiply them together, and we take all the numbers between 1 and the multiplication which do not divide either one of the two numbers. For instance, for 3 and 5 we get multiplication 15. We get 1,2, 4,7,8,11,13,14. What I want to talk about groups is that these groups have some properties:

groups have an operation. In the case of multiplicative group the operation is multiplication. This operation is associative. Also, the group is closed under this operation, which means if we take two elements in the group and you apply this operation to them, the result will stay in the group, you can’t get out of the group. The operation is multiplying mod: we multiply two numbers and we take the remainder modulo 15. This group has an identity element 1. The property of the element is that one time all of the elements in the group is the same element. And another property of the group is inverse. If we multiply a number with its inverse, we get 1. So the inverse of 4 for example is 4. A very important parameter in this group is the order of the group ɸ. This is the number of elements in the group. It always equal to (p-1) \* (q-1). So in our example there are 8 elements (5-1) \* (3-1) = 8. You cannot go from n to ɸ without knowing p and q. One of the tricks which makes very impressive and powerful is that you can do all sorts of operations without knowing ɸ, but for some operations you must know ɸ. You cannot get from n to ɸ or to p and q, without factoring n, which is still very difficult to do.

Of course p and q will be very large. So what can I do if I have multiplication? I can create another operation which I call exponentiation. And this works like in regular math. Exponentiation is closed. However, I can raise something to the power of any number. I can do 2 to the power of 3 even though 3 is not in the group. So just like the group has an order, each element has an order. The order means how many times do I exponentiate it until I get the 1. Every element has to have an order. Eventually, we will run out of the elements in the group (it is closed). Pherma little theorem states that the order of each elements always divides ɸ. That means that if I take an element and raise it to the power of ɸ, I will get 1. Now let’s see how it connects to RSA. RSA has public key and private key. The public key is used for encryption and the private key is used for decryption. The public key in RSA is <n,e>. The private key is either d or it can be <p, q>. The encryption key needs to be something which does not divideɸ . It can be a number which is not in Z\*. We are going to calculate d such that d=1 mod ɸ. How do I calculate d? using Xgcd. Now I want to encrypt a message. The message m has to be in Z\*15. How do I encrypt it? I raise it to the power of e. How do I decrypt it? I do C to the power of d modulo n.

Elementary Operation of RSA

* Choose an **encryption key *e*** and a   
  **decryption key *d*** s.t. *e∙d=*1 (mod (*p*-1)∙(*q*-1))
* To **encrypt** a given message *m*:  
   *C=me*(mod *n*)
* To **decrypt:** *M*=*Cd*(mod *n*)=*med*(mod *n*)=*m*1(mod *n*)=*m*
* Example from ℤ\*15:   
   3*∙*11*=*1 (mod 8), 23=8 (mod 15), 811=2 (mod 15)

My encryption key is 3, my decryption key is 11 in this example.

This is not a crypto course. Next week we will talk about exponentiations and side channels in RSA.

See also (articles in moodle):

1. John Wiley and Sons Chichester , "Overview about Attacks on Smart Cards by Wolfgang Rankl, Munich", 3rd edition at John Wiley and Sons in September 2003.
2. Thomas Popp, "An Introduction to Implementation Attacks and Countermeasures", Graz University of Technology, Institute for Applied Information Processing and Communications (IAIK) Graz, Austria.

References

[1] , Paul C. Kocher , "Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems", Cryptography Research, Inc.

[2] J.-F. Dhem, F. Koeune, P.-A. Leroux, P. Mestre , J.-J. Quisquater and J.-L. Willems, "A practical implementation of the timing attack", June 15, 1998.

[3] Burt Kaliski, "The Mathematics of the RSA Public-Key Cryptosystem", RSA Laboratories.