Great Ideas in Computing

University of Toronto CSC196
Winter/Spring 2019

Week 11: November 30-December 4 (2020)

Announcements

- Today and next week will be our last meetings. There will be a tutorial on Monday. Good place to ask questions on Assignment 4 as well as any of the previous assignments or quizzes.
- I moved the Assignment due date to Wednesday, December 9 (11 AM). Of course you can submit before the due date. However, we will only accept late assignments (with 5% penalty per 24 hours) for up to 48 hours. Regrade requestss must be made by Monday, December 14.
- We have the opportunity to have an additional class (for "make-up Monday") on Thursday, the 10th. I suspect most students are busy submitting end of term work so we will not take that opportunity.
- Marta and I will be busy grading and submitting grades. We will try to get the grading done as fast as we can. Feel free to email me if you would like to talk about anything to do with the course or even about future CS courses.
- I hope everyone did well on the quiz. I apologize that a couple of people had trouble submitting.
  Quick poll: Was the test appropriate?

A slight detour

ASIDE: I came across a 2014 Nw York Times artice (and then followed by a book) “To Siri with Love”. I recommend reading the artcle: https://www.nytimes.com/2014/10/19/fashion/how-apples-siri-became-one-autistic-boys-bff.html

This is also related to the movie “Her” .

The article discusses the role of Siri with regard to her then 13 year old son with autism. One can read this article and ask about the role of such electronic personal assistants. The article gathered lots of acclaim and also criticism.
Is Siri a great idea?

Where have we been and where are we going

- We ended last Friday, November 27 on slide 23 of the Week 10 slides. Following our discussion of the P vs NP issue, the $P \neq NP$ conjecture led us to our current discussion of complexity based cryptography.
- Today, we will finish up our discussion of complexity based cryptography. I think everyone should be aware that complexity based cryptography is essential for internet commerce.
- In our remaining time, I would like to very briefly mention some of the many other great ideas that have made computing so pervasive. As I have mentioned, when I previously taught a variant of CSC196, it was called SCI199 and it was a full year course. So we had much more time to discuss a variety of ideas.
- I will mainly be mentioning "older" great ideas that like so many great ideas we can't imagine any other way things could have been done.

Quick review of slides on secure shared secret key session

In this setting, two people called A and B (sometimes referred to as Alice and Bob) have been able to share *secret key* (e.g., a secret string of bits) and will use that secret key to communicate over an insecure channel. This insecure channel can be observed or perhaps even modified by an adversary.

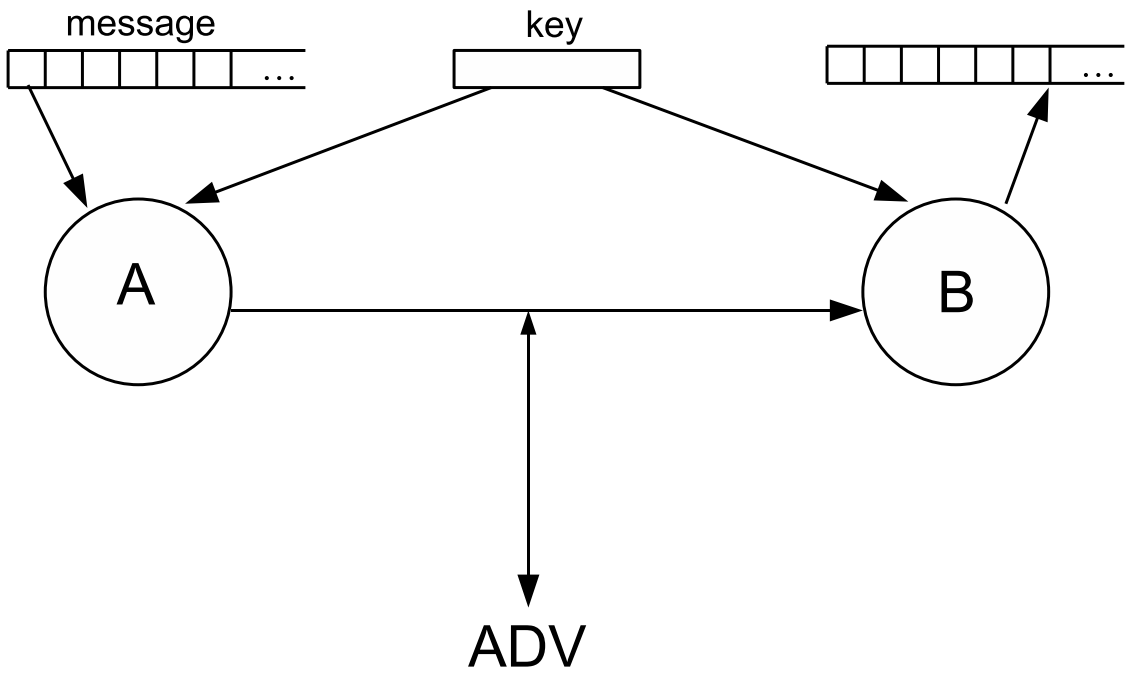

Figure: One-way communication. Figure taken from Rackoff notes

Shared-secret key session continued

An important consideration is how powerful is the adversary.
To do things reasonably carefully, we would probably need a full graduate course on cryptography. It is difficult enough to develop the main ideas even assuming that the adversary can only eavesdrop so lets make that assumption.

In a one-way session, A has an m bit message $M = M_1 M_2 \dots M_m \in \{0,1\}^m$. (For simplicity, we are assuming that the message and the secret key have been reresented as a binary strings but this is not essential.) The message is called the *plain text*.

In the shared secret key setting we are assuming the A and B have agreed upon a secret key n bit key $K = K_1 K_2 \dots K_n \in \{0,1\}^n$.

A will encode his message by a function $ENC : \{0,1\}^m \times \{0,1\}^n \to \{0,1\}^*$. Here we are using the $*$ to suggest that the encoded message length can depend on the plain text message. Why not a fixed length independent of the message?

Shared-secret key session continued

An important consideration is how powerful is the adversary.
To do things reasonably carefully, we would probably need a full graduate course on cryptography. It is difficult enough to develop the main ideas even assuming that the adversary can only eavesdrop so lets make that assumption.

In a one-way session, A has an m bit message $M = M_1 M_2 \dots M_m \in \{0,1\}^m$. (For simplicity, we are assuming that the message and the secret key have been reresented as a binary strings but this is not essential.) The message is called the *plain text*.

In the shared secret key setting we are assuming the A and B have agreed upon a secret key n bit key $K = K_1 K_2 \dots K_n \in \{0,1\}^n$.

A will encode his message by a function $ENC : \{0,1\}^m \times \{0,1\}^n \to \{0,1\}^*$. Here we are using the $*$ to suggest that the encoded message length can depend on the plain text message. Why not a fixed length independent of the message? The encoded message is called *the cypher text*.

Shared-secret key session continued

B will decode the cypher text by a function
$DEC : \{0,1\}^* \times \{0,1\}^n \to \{0,1\}^m$.

What properties do we want from the exchange?

Shared-secret key session continued

B will decode the cypher text by a function
$DEC : \{0,1\}^* \times \{0,1\}^n \to \{0,1\}^m$.

What properties do we want from the exchange?

- **Privacy:** The adversary should not learn anyything "signifiicant" about the plain text. What does the adversary know in advance about the plain text?
- **Correctness:** B should be able to correctly decode the message. That is $DEC(ENC(M,K),K) = M$ for all M and K.

We might ask what is *perfectly secure session*? In the Rackoff notes #0, there are three equivalent definitions. Let's just use the first one. For a given plain text message M, a uniformly random key K induces a distribution D_M on the cypher texts. The session is perfectly private if the distribution D_M does not depend on M.

Is a perfectly secure session attainable?

When is a perfectly secure session attainable?

Fact: A Perfectly secure session is attainable if and only if $|M| \leq |K|$.

When $|M| \leq |K|$, a *one-time pad* provides a single perfect secure session. A one-time pad is defined as follows:

$ENC(M, K) = E_1 E_2 \ldots E_m$ where $E_i = M_i \oplus K_i$ for $1 \leq i \leq m$ and $DEC(E, K) = E_1 \oplus K_1, \ldots, E_m \oplus K_m$.

Note that $\oplus$, the exclusive OR, flips a bit.

Warning: Never use an old key for a new purpose. For example, we cannot securely send two m bit messsages with the same m bit key.

So how are we going to continually generate random private keys (or long keys that can be partitioned into session keys) for different people to communicate? We cannot assume people can get together physically and even so how can they generate truly random strings of bits?

Complexity based assumptions; public key cryptography

The one-time pad does not need any assumptions and an adversary can have unlimited computational power and still cannot gain any information from a one-time pad. But as we noted, a one-time pad is not a very practical solution especially for frequent transactions in e-commerce.

The major application of *public key cryptography* is to enable key exchange. For public key cryptography (and almost all cryprographic applications) we will need complexity assumptions stronger than (but still widely accepted) $P \neq NP$. To make public key systems practical we will also need some sort of trusted public key infrastructre.

We will just discuss one well known public key system, *RSA*, which is based on the assumption that factoring large integers is hard even in some average sense (rather than worse case sense). This is a much stronger assumption than $P = NP$ since $P = NP$ would allow us to factor integers in polynomial time.

The basic idea of public key encryption

Public key encryption was introduced by Diffie and Hellman, and a particular method (RSA) was created by Rivest, Shamir and Adelman.

The basic idea is that in order for Alice (or anyone) to send Bob a message, Bob is going to create two related keys, a public key allowing Alice to send an encrypted mesasage to Bob, and a private key that allows Bob to decrypt Alice's message.

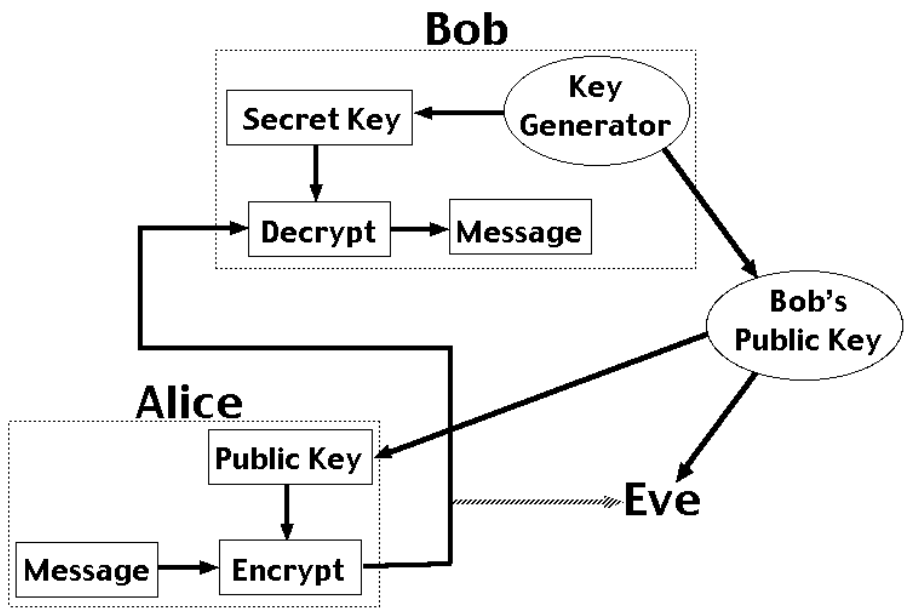

Figure: Diagram of public key encryption. Figure taken from Paul Johnston notes

The RSA method

Bob wants to generate two keys, a public key e, N and a private key d. The claim is that it is hard on average to find d given e and N. Bob chooses $N = p \cdot q$ for two large primes p, q (which for defining "on average" may satisfy some constraint).

Bob will choose the public e such that $gcd(e, \phi(N)) = 1$ where $\phi(N) = \phi(pq) = (p-1)(q-1)$. $\phi(N)$ is called the Euler totient function which is equal to the number integers less than N that are relatively prime to N. $gcd(a, b) = 1$ means that a and b are relatively prime (i.e. have no common proper factors).

Alice encodes a message M by computing M^e mod N.

Hiding some mathematics, BOB can compute a d such that $de = 1$ mod $(p-1)(q-1)$ since Bob knows p and q. But without knowing p, q, finding d becomes computationally difficult.

Hiding some more mathematics, it will follow that $M^{de} = M \pmod{N}$ for any message M. That is, Bob decrypts a cypher text C by the function C^d mod N.

What mathematical facts do we need to know.

The main mathematical facts are :

1. There are sufficiently many prime numbers in any range so one can just randomly try to diffent numbers and test if they are prime.
2. $a^{\phi(N)} = 1 \bmod N$ for any a such that $gcd(a, N) = 1$ As a special case, $a^{p-1} = 1 \bmod p$ for any prime p and a not a multiple of p. So we have $M^{(p-1)(q-1)} = 1 \bmod N$.
3. If $gcd(a, b) = 1$ then there exists s and t such that $sa + tb = 1$. In the RSA algorithm, we can let $a = e$ and $b = (p-1)(q-1)$. Then s will become the d we need for decryption. That is $de + t(p-1)(q-1) = 1$.
4. It follows then that $M^{de} = M^{1-t(p-1)(q-1)} = M \cdot M^{-t(p-1)(q=1)} = M \bmod (p-1)(q-1)$.

What computational facts do we need to know?

1. The extended Euclidean algorithm can efficiently compute an s and t such that $sa + tb = gcd(a, b)$
2. a^k mod N can be computed efficiently for any a, k, N.
3. We can efficiently determine if a number p is prime.

In practice, public keys e are chosen to be reasonably small so that encryption can be made more efficient.

Note that we have been assuming that an adversary EVE (i.e., is just eavesdropping) and not changing messages. That is, EVE just wants to learn the message or something about the message. If EVE could change messages then EVE could pretend to be BOB. So one needs some sort of a public key infrastructure.

Note that if EVE knows that the message M was one a few possibilities, then EVE can try each of the possibilities; that is compute $M^e mod N$ for each possible M to see what message was being sent. So here is where randomness can be used. We can pad or interspers random bits in the plain text M so that the message being sent becomes some one of many random messages M'.

WARNING: Real world cryptography is sophisticated

Complexity based cryptography requires careful consideration of the definitions and what precise assumptions are being made.

Complexity based cryptography has led to many important practical protocols and there are a number of theorems. Fortunatley, many complexity assumptions turn out to be equivalent.
In the Rackoff notes, the following theorem is stated as the fundamental theorem of cryptography. (To make this result precise, one needs precise definitions which we are omitting.)
Theorem: The following are equivalent:

- It is possible to do "computationally secure sdssions"
- There exists pseudo-random generators; that is, create strings that computationally look random)
- There exist one way functions f; that is functions such that $f(x)$ is easy to compute but given $f(x)$ it is hard to find a z such that $f(z) = f(x)$
- There exist computationally secure digital signature schemes.

The discrete log function

RSA is based on the assume difficulty of factoring. Another assump;tion that is widely used in cryptography is the discrete log function. Again, we need some facts from number theory.

Let p be a large prime.

- $\mathbb{Z}_p^*$ denotes the set of integers $\{1, 2, \ldots, p-1\}$ under the operations of $+, -, \cdot \mod p$ is a *field*. In particular, for every $a \in \mathbb{Z}_p^*$, there exists a $b \in \mathbb{Z}_p^*$ such that $a \cdot b = 1$; i.e., $b = a^{-1} \mod p$.
- Moroever, $\mathbb{Z}_p^*$ is *cyclic*. That is, there exists a $g \in \mathbb{Z}_p^*$ such that $\{1, g, g^2, g^3, \ldots g^{p-2}\} \mod p = \mathbb{Z}_p^*$. Recall, as a special case of the Euler totient function, $a^{p-1} = 1 \mod p$.

The assumption is that given $(g, p, g^x \mod p)$, it is computationally difficult to find x. This is another example (factoring can also be an example) of a *one-way function*.

A pseudo random generator

We started off our discussion of complexity based cryptpgraphy by noting that randomness is essential. We have also noted that it is not clear (or at what cost) one can obtain strings that "look like" truly random strings.

A pseudo random generator G is a *deterministic* function $G : \{0,1\}^k \to \{0,1\}^\ell$ for $\ell > k$. When ℓ is exponential in k, G is called a pseudo random function generator. For now, lets even see how to be able to have $\ell = k + 1$.

The random input string $s \in \{0,1\}^k$ is called the seed and the goal is that $r = G(s)$ should be "computationally indistinguishable" from a truly random string in $t = \{0,1\}^\ell$. This means that no polynomial time algorithm can distinguish between r and t with probability better than $\frac{1}{2} + \epsilon$ for any $\epsilon > 0$. (Here I am being sloppy about the quantification but hopefully the idea is clear.)

A pseudo random generator continued

On the previous slide there was a claim that having a pseudo random generator is equivalent to having a one-way function.

How can we use (for example, the assumption that the discrete log function is a one-way function) to construct a pseudo random generator with $\ell = k + 1$.

The Blum-Micali generator. Assumming the discrete log function is a one-way function then the following is a pseudo random generator:

Let x_0 be a random seed in $\mathbb{Z}_p^*$ by interpeting $(s_1, \ldots, s_k)_2$ as a binary number mod p. Let $x_{k+1} = g^{x_k} \bmod p$. Define $s_{k+1} = 1$ if $x_k \leq \frac{p-1}{2}$.

Manual Blum won the Turing award for his contributions to cryptography and Silvio Micali (along with Shafira Goldwasser) won the Turing award for *interactive zero knowledge proofs*.

Next week: Some great ideas we did not discuss

There are many great ideas that we have not had time to discuss. In our remaining time, I want to at least mention some great ideas. Namely, I want to briefly mention the following great ideas.

- Programming languages in contrast to machine code and assembly languages.
- Operating systems in contrast to submitting jobs individually
- Graphical user interfaces (GUIs) (and the mouse) in contrast to a command line interface
- Packet routing in contrast to virtual circuit routing and the internet
- Personal computers in contrast to main frames

If all of these seem like "duh, what else would you do", that is perhaps the best evidence of a great idea; that is when it becomes so common place, that it is hard to imagine anything else. But these great ideas often came with resistance.