CS 6160 Cryptology Lecture 17: Digital Certificates & Advanced Topics : Zero-Knowledge Proofs

Maria Francis

November 17, 2020

Digital Certificates

- Motivation: secure distribution of public keys.
- PKC was to securely distribute keys and now we are looking the same problem again? Not really!
- Once a single public key, belonging to a trusted party, is distributed in a secure fashion, then that key can be used to securely distribute arbitrarily many other public keys.
- Digital certificates is the key idea here : a signature binding an entity to some *pk*!
- How does it work?

Digital certificates

$\xleftarrow{\text{cert}_{C\to B}}$

Bob

Carol

(pk_B, sk_B)

(pk_C, sk_C)
Carol knows pk_B
is Bob's key

$\text{cert}_{C\to B} := Sign_{sk_C}(\text{'Bob's key is } pk'_B)$.
Usually it is URL of Bob's website or full name and email address not the name Bob.

Digital certificates

Bob

$$\xrightarrow{(pk_B, \mathrm{cert}_{C \to B})}$$

Alice

Alice knows pk_C and trusts Carol.

Alice now knows that Carol has signed that "pk_B is Bob's key" and Alice trusts Carol so she accepts pk_B as Bob's legitimate public key.

Digital certificates & PKI

- All communication is happening over an insecure and unauthenticated channel.
- Even if $\mathcal{A}$ interferes with $(pk_B, \mathrm{cert}_{C \to B})$, he cannot create a valid certificate linking Bob to the other public key pk'_B unless Carol's private key is compromised or Carol cannot be trusted.
- How does Alice learn about pk_C?
- How does Carol know pk_B is Bob's public key?
- How does Alice know how to trust Carol?
- All this defines various different public key infrastructure (PKI).

Single Certificate Authority

- The simplest PKI that assumes a single certificate authority (CA) who is trusted by everybody and issues certificates for everyone's public key.
- CA could be a company who certifies public keys, a government agency, or a department within an organization.
- We should make sure we get a legitimate copy of the CA's pk_{CA} – this should be distributed over an authenticated channel.
- How can that happen?
  - If CA is a dept within a company then do it physically.
  - For scalable scenarios typically it is bundled with software like web browsers.
  - The browser automatically verifies certificates as they arrive.
  - Typically, browsers have *pks* of *multiple* CAs hardwired into their code.

Multiple Certificate Authority

- How does CA issue a certificate to a particular Bob? It depends on CA. Bob can show up in person and ask for a certificate, for example.
- Single CA is simple but not practical. Everyone may not find one CA's verification process enough and it is also a single point of failure/attack.
- One solution: multiple CAs. Bob can get multiple certificates from multiple CAs (just that it will be more expensive for Bob and time consuming too!)
- Alice has to be careful which CA of the multiple ones trusts Bob. What if the less trustworthy CA attests Bob?
- Usually all CAs are given equal trustworthiness in the configuration. It is left to the user to change the configuration to give more importance to established, reputed CAs.

Certificate Chains

- Carol is a CA that issues a certificate for Bob.
- Bob in turn issues a certificate for Alice.

$$\mathrm{cert}_{B \to A} := \mathit{Sign}_{sk_B}(\text{`Alice's key is } pk'_A)$$

- Alice wants to communicate with another person Dave who trusts Carol and knows Carol's pk_C.
- Alice sends Dave:

$$pk_A, \mathrm{cert}_{B \to A}, pk_B, \mathrm{cert}_{C \to B}.$$

- Dave first sees that since he trusts Carol that pk_B is indeed Bob's key and from $\mathrm{cert}_{B \to A}$ that pk_A is indeed Alice's public key.

Certificate Chains

- $\mathrm{cert}_{C \to B}$ means now Bob holds pk_B and *Bob is trusted to issue other certificates.*
- This chain can be extended to arbitrary length.
- This is called delegation, Carol delegating the ability to issue certificates to Bob.

Web of Trust Model

- Fully distributed model with no central points of trust.
- A variant is Pretty Good Privacy (PGP) : email encryption software for distribution of public keys.
- Anyone can issue certificates and each user makes a decision how much trust to place in each certificate.
- Users are expected to collect both public keys of other parties as well as certificates on their own public key.

Web of Trust Model

- For PGP, there were key-signing parties where the users gave each other authentic copies of public keys.
- At these meetings you can check physical evidence like a driver's license.
- Decentralized model is attractive because no central authority but where security is critical it does not really work. For eg: for a bank transaction do you trust a person you met at a party?
- There is also the issue of how long should a certificate be valid and also revocation of a certificate like when the private key is stolen, etc.

Conclusion on basics

- We have come to an end to all the basics of cryptography that we planned to cover.
- There are many many more things that fall under the blanket of basics of the area. Some of them are:
  - Practical constructions of hash functions
  - Pseudo Random Functions more formally
  - Hash function families more formally
  - Cryptanalysis more examples (we have interesting student presentations!)
  - Security definitions more details
  - Computational Number Theory: Algorithms for factoring, computing primes and discrete logs.
  - More details of Hybrid Encryption schemes
  - Signatures from Discrete-Log problem and many many more.

Zero-Knowledge Proofs

- A landmark cryptographic proof/protocol by Shafi Goldwasser, Silvio Micali, and Charles Rackoff (1989).
- Interactive proofs that reveal nothing other than the validity of assertion being proven.
- They got the first Gödel prize for this work.
- Oded Goldreich, Silvio Micali, and Avi Wigderson showed that one can have a ZKP for the NP-complete graph coloring problem with 3 colors.
- Since every problem in NP can be efficiently reduced to this problem, this means that all problems in NP have zero-knowledge proofs.
- But there is a key assumption: "Existence of unbreakable encryption" since these protocols require encryption and that implies existence of OWFs.

Zero-Knowledge Proofs

- I will be following the slides of Yevgeniy Dodis and Chapter 4 of the textbook "O. Goldreich. Foundations of Cryptography - Volume I (Basic Tools)."
- The following winter school lectures are also very useful:`https://cyber.biu.ac.il/event/the-9th-biu-winter-school-on-cryptography/`
- The material we cover is the basics of ZKP and there is a lot that we do not cover including the versatile applications of these proofs.
- The idea is to get a feel for the topic.

Motivation

- Revealing parts of secret without revealing the whole thing among mutually distrustful parties.
- Consider the following example:
  - All users in a system keep backups of their files encrypted using secret keys in a publicly accessible storage.
  - At some point Alice wants to reveal to Bob the clear text of some record in one of her files.
  - Alice can simply send Bob the clear text but how will Bob verify if it was indeed the record and not something Alice arbitrarily sent?
  - Alice could reveal the secret key but that would mean Bob gets to see everything.
  - The question is whether or not Alice can convince Bob that she has indeed revealed the correct record without yielding any additional knowledge.

Motivation - more formally!

- Let f be a OWP and b a hard-core predicate of f.
- Let Alice have a string x whereas Bob has only $f(x)$.
- Alice wants to reveal $b(x)$ to Bob without giving any information.
- If Alice just sends $b(x)$ to Bob then how will know Bob verify that Alice is not cheating?
- Or Alice could send x and $b(x)$ as well but that is revealing too much.
- We want to prove a statement S without yielding anything beyond its validity. Such proofs are called zero-knowledge.

Proofs – whatever convinces me! (Shimon Even)

- In mathematics proof is a fixed sequence consisting of statements that either are self-evident (axioms) or are derived from previous statements via self-evident rules.
- It is static. Proofs we consider are dynamic in nature i.e. they are established by interaction.
- A real-life example is withstanding a cross-examination in court which can yield a proof in law.
- Prover P wants to prove to Verifier V some statement S is true.
- The verification procedure is considered to be relatively simple, and the burden is placed on the prover.

Class *NP*

- Asymmetry between complexity of P's and V's task is captured by *NP*, the class of proof systems.
- Witness of S: string w s.t. V can check S is true using w.
- *NP*: where each true statement has a witness, and false statements do not have any.
- $\mathcal{L} \in NP$ can efficiently verify $w \in \mathcal{L}$ but w might be difficult to find!
- That is, coming up with a proof might be hard (unless *NP* is contained in *BPP*).

Class NP

A proof system for a language $\mathcal{L}$ is a poly-time algorithm V (verifier) s.t.

1. Completeness : *True statements have proofs*

$$x \in \mathcal{L} \Rightarrow \exists \; proof \text{ s.t. } |proof| \leq \mathrm{poly}(|x|)$$
$$\text{and } V(x, proof) = accept.$$

2. Soundness: *False statements have no proofs*

$$x \notin \mathcal{L} \Rightarrow \forall \; proof^* \;\; V(x, proof^*) = reject$$

Class *NP*

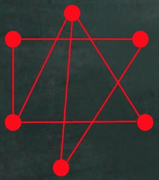
- *NP* is the class of languages with proof systems.
- Provers are often implicit in discussions of proofs and V is more explicit and the verification process typically has to be easy.
- Note that there is a distrustful attitude towards P in any proof system since no proof is needed if V trusts P.
- Soundness ensures no prover can trick the proof system and completeness is the ability of some P to convince V of true statements.

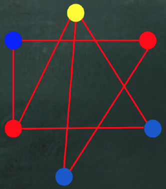
3-coloring of graphs

- A 3-coloring of a graph is an assignment of colors in say, $\{\text{Red}, \text{Blue}, \text{Yellow}\}$, to vertices s.t. no pair of adjacent vertices are assigned the same color.
- Proposition: 3-COL$= \{G : G$ is 3-colorable$\}$ is in NP.
- A language $\mathcal{L}$ is NP-complete if $\mathcal{L} \in NP$ and every language in NP reduces to $\mathcal{L}$.
- Theorem: 3-COL is NP-complete.

Examples

Three coloring is possible:





Three coloring is not possible:

What needs to be added?

- Classical *NP*-proofs are inherently non-zero-knowledge. V gains ability to prove $w \in \mathcal{L}$ to others.
- We allow for randomization: V can toss coins. This allows for V to err with small probability.
- Interaction: replace static proof with dynamic, all-powerful prover. Will interact with verifier and try to convince it that statement is true.

Interactive protocols

- Interaction between P and V is defined in the natural manner. The interaction is parameterized by a common input x.
- In interactive proof systems, x is the statement to be proved.
- Polynomially bounded: lengths of all messages & no of messages all $\mathrm{poly}(|x|)$.
- Alice and Bob are functions: $(x, \text{random coins}, \text{previous messages}) \mapsto (\text{ next message})$.
- Messages are typically the strings from the alphabet along with accept, reject, halt.

Interactive Proofs

- An interactive proof system for a language $\mathcal{L}$ is an interactive protocol (P,V) where
  1. V is poly-time computable.
  2. Completeness: If $x \in \mathcal{L}$ then V accepts $(P, V)(x)$ with probability 1.
  3. Soundness: If $x \notin \mathcal{L}$ then for every P^*, V accepts in $(P^*, V)(x)$ with probability $\leq 1/2$.
- *IP* is the class consisting of all languages having interactive proof systems.

Interactive proofs

Computationally unbounded

Poly-time computable
Random coins

Prover

m_1 →

← m_2

Verifier

Common parameter : x

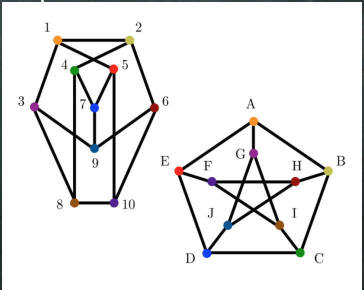
Class *IP*

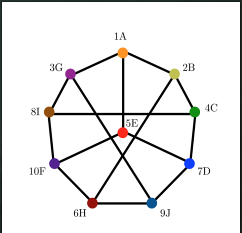
- Can reduce error probability in soundness to 2^{-c} with c repetitions, where c is a constant.
- *NP* have interactive proof systems in which both parties are deterministic (verifier never errs) and the communication is unidirectional (from P to V). $NP \subseteq IP$.
- If V is deterministic then *IP* collapses to *NP*.
- Interactive proofs generalize classical proofs.
- *IP* is likely to be bigger: $IP = PSPACE$.
- Combination of interaction and randomization has a huge effect: the set of languages which have interactive proof systems now jumps from *NP* to *PSPACE*.

Graph Isomorphism & Nonisomorphism

- $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ are called isomorphic ($G_1 \cong G_2$)if there exists a bijective function $\pi : V_1 \to V_2$ s.t. $(u, v) \in E_1$ iff $(\pi(u), \pi(v)) \in E_2$.
- π is called the isomorphism between the graphs and if no π exists we say the graphs are non-isomorphic.
- The corresponding languages :
  - $GI = \{(G_1, G_2 : G_1 \cong G_2\}$
  - $GNI = \{(G_1, G_2 : G_1 \ncong G_2\}$

Graph Isomorphism





Source: `https://algorist.com/problems/Graph_Isomorphism.html`

GI and *GNI*

- *GI* is in *NP* since isomorphism is a witness. It is unlikely to be *NP*-complete.
- *GNI* is considered to be neither in *NP* or *BPP*.
- Hard to check if no isomorphism exists.
- W.r.t *GI* the question we ask is : How can a *P* prove that two graphs are isomorphic without revealing the isomorphism? Zero-knowledge!
- W.r.t. *GNI* we ask: how can a *P* convince a *V* that two graphs are non-isomorphic? Note that here we do not have a witness!
- We will see the power of interactive proofs!

Interactive Proof for *GNI*

- Common parameter:
  $G_1 = (\{1, \dots, n\}, E_1), G_2 = (\{1, \dots, n\}, E_2)$
- Verifier V:
  1. Chooses $i \in \{1, 2\}$ randomly and a permutation π of $\{1, \dots, n\}$.
  2. It applies π on the ith graph to get

$$H = (\{1, \dots, n\}, \{(\pi(u), \pi(v) : (u, v) \in E_i\}).$$

     V is constructing randomly a graph isomorphic to the graph it chose.
  3. Sends H to P.
- Prover sends $j \in \{1, 2\}$ to the V.
  - If the two graphs are non-isomorphic then P can find which of the two graphs this graph is isomorphic to the graph he received from V and send the correct answer.
- V accepts iff $i = j$.

Proof of *IP*

- Completeness: If G_1 and G_2 are non-isomorphic, as P claims,
  - the graph V sends is isomorphic to only one out of the two graphs,
  - P should be able to distinguish *(not necessarily by an efficient procedure)* isomorphic copies of one graph from isomorphic copies of the other graph
  - P can always send the correct answer.
- Soundness: If G_1 and G_2 are isomorphic, then a random isomorphic copy of one graph will be distributed identically to a random isomorphic copy of the other graph and the probability that $j = i$ is $\leq 1/2$.

Power of *IP*

- Thus $GNI \subseteq IP$ though mostly likely GNI is not in NP.
- Note: it is essential that the prover not know the outcome of the verifier's internal coin tosses.
- Is it ZK? V knows what the answer of P will be in advance so it is learning nothing new!
- What if she chooses H in a carefully designed way? Then she learns which graph H is isomorphic to.

Interactive Proof for GI

- Common parameter:
  $G_1 = (\{1,\ldots,n\}, E_1), G_2 = (\{1,\ldots,n\}, E_2)$.
- Prover knows of a permutation from σ from G_1 to G_2.
- P chooses a random permutation π of $\{1,\ldots,n\}$.
- P applies π on G_1 to get
  $H = (\{1,\ldots,n\}, \{(\pi(u),\pi(v) : (u,v) \in E_1\})$ and sends this to V.
- V sends random $j \in \{1,2\}$ to P.
- If $j = 1$, P sends $\tau = \pi$ to V else he sends $\tau = \pi\sigma$.
  - σ ensures G_2 is transformed to G_1 and π ensures it gets transformed to H.
- V checks if $\tau(G_j) = H$!

Proof for *IP*

- Completeness: If G_1 and G_2 are isomorphic, H is isomorphic to G_1 and G_2 and so with τ he can check the isomorphism.
- Soundness:If G_1 and G_2 are not isomorphic, then any H is not isomorphic to at least one of them and since V choose j randomly the probability that P can get away with it is at most $1/2$.
- Is it ZK? V only learns a random graph is isomorphic to one of the inputs.
- But what if V did not choose j at random?

Definition for ZKP

An interactive proof system (P, V) for a language $\mathcal{L}$ is zero-knowledge if whatever can be efficiently computed after interacting with P on input $x \in \mathcal{L}$ can also be efficiently computed from x (without any interaction).

- Key point: This holds with respect to any efficient way of interacting with P, not just the way we have defined V.
- It is the property of the prescribed prover, it captures its robustness against attempts to gain knowledge by interacting with it.

Formal Definition for ZKP

- Let (P, V) be an interactive proof system for some language $\mathcal{L}$.
- We say that (P, V) or P is perfect ZK (PZK) if
  - for *every* probabilistic poly-time machine V^* there exists an PPT algo M^* s.t. $\forall x \in \mathcal{L}$ the following two RVs are identically distributed:
  - $\langle P, V^* \rangle_{(x)}$ and $M^*_{(x)}$

  M^* is called a simulator for the interaction of V^* with P.
- We require that for every V^* interacting with P.
- The fact that such simulators exist means that V^* does not gain any knowledge from P - ZK!
- In practical applications, we say expected poly-time for simulators, i.e. polynomial time on an average.

Complexity classes based on ZKP

- Every language in BPP has perfect ZK since P does nothing and V can verify with just common inputs.
- When the two RVs are computationally indistinguishable you get computational ZK (CZK). Less stringent than i.d.
- $BPP \subseteq PZK \subseteq CZK \subseteq IP$.
- If OWFs exist $CZK = IP$ and other two are strict inclusions.

Re-looking our examples

- Were GI, GNI protocols we discussed ZKP? They had very simple simulators against honest Vs.
- Honest-verifier zero-knowledge, a weaker notion that works for a prescribed verifier and not any V.
- But this is useful and non-trivial and public-coin protocols that are HVZK can be transformed into similar protocols that are ZK in general.
- The GI interactive proof we saw is PZK but needs complicated analysis.
    - Difficult part is to simulate the output of an efficient V that deviates arbitrarily from the specified program.

Re-looking our examples

- The *GNI* protocol : not ZK unless $GNI \in BPP$.
- A cheating verifier can construct an arbitrary graph H and learn whether or not H is isomorphic to the first input graph by sending H as a query to the prover.
- It is HVZK!
- We can modify the construction to obtain a ZKP for *GNI* by having
  - V prove to P that she knows the answer to her query graph.
  - I.e., that she knows an isomorphism to the appropriate input graph
  - P answers the query only if he is convinced of this claim.

ZK Proof for 3-COL

1. Randomly permute colors

Prover

Colors in locked boxes $\longrightarrow$

Verifier

Graph G

ZK Proof for 3-COL

Prover

(i, j)

Verifier

Graph G

Pick a random edge (i, j)

ZK Proof for 3-COL

Prover

Send keys for boxes of i, j →

Verifier

Graph G

Open boxes

Accept if colors are different

ZK Proof for 3-COL

- Completeness: graph is 3-colorable means V accepts with certainty.
- Soundness: graph is not 3-colorable means $\forall$ P, V rejects with probability $\geq 1/|E|$.
  - Any content placed in the boxes must be invalid on at least one edge.
- Zero-Knowledge: graph is 3-colorable means V sees two random distinct colors, no knowledge!

ZK Proof for NP Complete

- Since 3-COL is *NP*-complete, intuitively this implies the existence of a zero-knowledge proof system for every language in *NP*. The proof is available in Goldreich's textbook and is not that difficult to understand.
- Confidence in the validity of the claim can be increased by sequentially applying the foregoing proof sufficiently many times.
- This is because sequential composition of zero-knowledge proofs is also zero-knowledge though zero-knowledge is not preserved in parallel composition.

How to implement locked boxes?

- Commitment schemes. Digital analogue of locked boxes.
- Two-phase (Commit, Reveal)two-party protocol through which Sender can commit itself to a value where two conflicting requirements are satisfied.
  1. Secrecy (or hiding): After Commit phase, receiver does not gain any knowledge of the sender's value.
  2. 2. Unambiguity (or binding): Sender cannot change value after Commit phase

Commitment Schemes

- The receiver should receive the sender's value after reveal phase and there can be only one legal opening, i.e. it should be unambiguous.
- It can either be: computationally hiding and perfectly binding or vice-versa but not perfectly hiding and perfectly binding.
- Existence of OWFs $\Rightarrow$ bit commitment schemes.
- Simpler constructions possible from factoring, discrete log problem.

Constructions from OWFs

- From injective OWFs.
- Remember that every OWF $f(x)$ can be modified to have a hard-core bit $h(x)$ (Goldreich-Levin Theorem).
- To commit to a bit b, sender picks a uniformly random x and sends,

$$(h, f(x), b \oplus h(x)).$$

Note it is the function h and not its value at x.

- Reveal phase: Sender just sends x.
- Receiver computes $f(x)$ and verifies.
- Perfect binding: since f is injective.
- Computational hiding: Since $h(x)$ is hard-core bit recovering it from $f(x)$ has only probability negligibly better than a random guess.

Constructions from Discrete Log

- Prime order p group with generator g.
- Sender commits to $x \in \{0, \ldots, p-1\}$ by publishing $c = g^x$.
- From discrete log problem you have, it is computational hiding.
- Sender cannot provide another x' s.t. $g^{x'} = c$ perfectly binding.
- Not the best scheme since it is not secure w.r.t. the CPA experiment.
- Pedersen commitment scheme : very popular. A perfectly hiding commitment scheme with binding based on discrete-log.

The Way Forward

- ZK not preserved under parallel repetition.
- Many rounds can it be replaced by constant rounds with negligible error?
- Non-interactive ZK.
- Practical protocols and efficiency issues
- Many of these are taken care of by Σ-protocols!

Applications of ZKP