The problem statement

• Let $L$ be some language (set of words), let $x$ be an (encrypted) value

• How to prove that $x \in L$ without giving out any additional information?
  
  \* $x$ is positive? $x$ is a full square? $x$ is prime?

• General: how to prove that “I know that $x \in L$”

• After decrypting, verifier would see $x$ and could test that $x \in L$ but it would give more information than is often necessary
Usage examples

- Familiar scenario: authentication

- Private key: \( x \), public key: \( g^x \)

- I want to prove you that I know the discrete logarithm of \( g^x \)

- Without revealing \( x \) itself!

You already saw this scenario (identification schemes), but these schemes were not zero-knowledge
What is knowledge?

- Hard to define - it is easier to define what is \textit{gain of knowledge}.

- I tell you \(1 + 1 = 2\). Do you gain knowledge?

  ★ Most of you don’t.

- I tell you the factors of \(2^{241} - 1\). Do you gain knowledge?
Minimizing gain of knowledge

- I prove you that I know the factors of $2^{2^{41}} - 1$, without revealing them.

- I prove that two graphs $G_1$ and $G_2$ are isomorphic without revealing the isomorphism.
  
  ★ Graph isomorphism is a well-known hard problem

- In general: I convince you that I know something, without you getting to know anything else but that I know this something
  
  ★ $\approx$ zero-knowledge.
Knowledge ≠ Information

**Information:** You are revealed an unknown object.

- Factors of $2^{241} - 1$: no new information
- Properties of information are studied in information theory

**Knowledge:** You are revealed results of calculations on a publicly-known object that you cannot derive by yourself.

- Factors of $2^{241} - 1$: probably new knowledge
- Factors of a randomly generated 1024-bit integer: new knowledge, assuming that factoring is hard
Zero-knowledge: Intuition

- We talk about ZK protocols between verifier $V$ and prover $P$

- Big intuition: Zero-knowledge is a property of prover $P$:
  - Given a common input $x$ with prover $P$, whatever you can calculate, based on the interaction with $P$, you can calculate based on $x$ alone.

- I.e., you can simulate $P$.

- Proof system: $P$ still manages to convince you that $x \in L$. 

---

T-79.159 Cryptography and Data Security, 12.03.2003   Lecture 7: ZK and Commitments, Helger Lipmaa
Preliminaries

- For formal definition of ZK, one must define an *interactive proof system* (IP system)

- IP system consists of two interactive machines that both have private
  - (read-only) input, (read-only) random string, read-write working space, (write-only) output

- Machines can also communicate by sending messages
Preliminaries: Interactive Protocols

- A protocol takes several steps of communications, where in every step one participant sends a message to another one.

- An interactive protocol IP is a pair \((P, V)\), where at every step one participant decides, based on the previous communication, private and common inputs, and on the random string what would be the next input.

- We assume that \(P\) is computationally unbounded.

- \(V\) is computationally bounded.
Interactive proof system

Language $L$ has an interactive proof system if there is such an interactive machine $V$, so that

- $\exists P$, so that $\forall x \in L$, $V$ “accepts” the common input after the IP $(P, V)$ with probability $\geq 2/3$

- $\forall P^*$, where $(P^*, V)$ is an IP: For all $x \not\in L$, the probability that $V$ “accepts” is $< 1/3$

- (Probabilities are taken over the coin tosses of $P, V$)

- Let IP be the set of languages that have IP proofs
Example 1: Quadratic Residues

- Recall that $\mathbb{Z}_n^* = \{0 < x < n : \gcd(x, n) = 1\}$.

- Quadratic residues modulo $n$:
  \[ \text{QR}(n) := \{x \in \mathbb{Z}_n^* : (\exists y) y^2 \equiv x \mod n\}, \]
  elements that have a square root modulo $n$

- Quadratic nonresidues:
  \[ \text{QNR}(n) := \{x \in \mathbb{Z}_n^* : (\not\exists y) y^2 \equiv x \mod n\}. \]
Example 1: Quadratic Residues

- For prime \( n \), establishing whether \( x \in QR(n) \) will be trivial

- For RSA modulus \( n = pq \), establishing whether \( x \in QR(n) \) is equivalent to factoring \( n \)

- Quadratic Residuosity Assumption (QRA): For non-prime \( n \) and random \( x \in \mathbb{Z}_n \), establishing whether \( x \in QR(n) \) is hard

- We will assume \( n \) is not prime

  \* QRA: \( x \in \mathbb{Z}_n \) is hard
Example 1: IP for QNR(n)

Parameter \( k \) and common input \((x, n)\), where \( x \in \text{QNR}(n) \).

- **V** generates \( k \) random numbers \( z_i \leftarrow_R \mathbb{Z}_n^* \) and \( k \) random bits \( b_i \), and sends to **P** the tuple 
  \[
  (w_1, \ldots, w_k),
  \]
  where \( w_i \leftarrow x^{1-b_i} \cdot z_i^2 \mod n \).

- **P** sends to **V** a tuple  
  \[
  (c_1, \ldots, c_k),
  \]
  where \( c_i \leftarrow 1 \) iff \( w_i \in \text{QR}(n) \).

- **V** accepts that \( x \in \text{QNR}(n) \) iff \( b_i = c_i, \forall i \)
Correctness of example 1

- If \( x \in \text{QNR}(n) \) then \( w_i = x^{1-b_i} \cdot z_i^2 \in \text{QR}(n) \iff b_i = c_i \). Since an omnipotent \( P \) can always establish whether \( w_i \in \text{QR}(n) \), she can also return the correct \( b_i \). Therefore, she can make \( V \) to accept with the probability 1.

- If \( x \in \text{QR}(n) \) then \( w_i \) will be a randomly chosen quadratic residue, independently of the value of \( b_i \). Thus the best strategy for \( P \) would be to guess \( b_i \) randomly, which means that the probability that \( b_i = c_i \), \( \forall i \), is \((1/2)^k\)

  - Enlarging \( k \) will decrease this probability but will also make the protocol less efficient.
Example 2: Graph Nonisomorphism

- Recall: A graph $G$ is a set of vertices $V(G)$ together with some set $E(G) \subseteq V(G) \times V(G)$ of edges.

- Two graphs $G_1$ and $G_2$ are isomorphic if there exists a bijection $\pi : V(G_1) \rightarrow V(G_2)$, s.t.

  $$(v, w) \in E(G_1) \iff (\pi(v), \pi(w)) \in E(G_2).$$

  Otherwise $G_1$ and $G_2$ are nonisomorphic.

- Define $\text{GNI} := \{(G_1, G_2) : G_1$ and $G_2$ are not isomorphic\}.
Example 2: Graph Nonisomorphism

Are these two graphs nonisomorphic?
Example 2: Graph Nonisomorphism

No! They are isomorphic: we can show an isomorphism (mapping between the nodes).

But how to show nonisomorphism? (How to convince verifier that graphs are nonisomorphic, without sending too much information?)
Example 2: Graph Nonisomorphism

- A problem is in \textbf{NP} if we know a short witness
  - For graph isomorphism (GI), we can show \( \pi \)
  - Thus \( GI \in \textbf{NP} \)

- It is not known whether \( GNI \in \textbf{NP} \)

- We will show that \( GNI \in \textbf{IP} \)
IP for GNI

Common input \((G_1, G_2)\). Iterate the next step for \(i = 1 \ldots k\):

- \(V\) chooses a random \(\alpha_i \leftarrow R \{1, 2\}\), and a random graph \(G'_{\alpha_i}\) from the set of graphs that are isomorphic to \(G_{\alpha_i}\). She sends \(G'_{\alpha_i}\) to \(P\)

- (Omnipotent) \(P\) finds a graph \(G_{\beta_i}\), s.t. \(G_{\beta_i}\) and \(G'_{\alpha_i}\) are isomorphic, and sends \(\beta_i\) to \(V\)

  ✴ Intuition: \(P\) can guess \(\alpha_i\) iff graphs are nonisomorphic

\(V\) accepts iff \(\beta_i = \alpha_i, \forall i\)
Correctness of example 2

- When \((G_1, G_2) \in \text{GNI}\):
  - \(P\) can distinguish isomorphic copies of graph \(G_1\) from isomorphic copies of \(G_2\); then \(V\) accepts with probability 1

- When \((G_1, G_2) \notin \text{GNI}\):
  - An isomorphic copy of \(G_1\) is always an isomorphic copy of \(G_2\). Thus the best strategy for \(P\) is to toss a coin, and hence the cheating probability is again \((1/2)^k\).
Back to ZK and formal definition

- Let us have an interactive proof system \((P, V)\)

- \(\text{view}^P_V(x)\) — view of \(V\) when interacting with \(P\) on common input \(x\)

  \* \(\text{view}^P_V(x)\) is equal to the concatenation of all messages sent in this protocol, prefixed with all random coin tosses of \(V\)

- In the previous protocol:

  \* \((\alpha_1, \ldots, \alpha_k) \parallel (G'_1, \beta_1, \ldots, G'_k, \beta_k)\)
Formal definition (First try)

Definition. Let \((P, V)\) be an IP system for language \(L\). \((P, V)\) is (perfect) zero-knowledge if for every machine (probabilistic polynomial-time) machine \(V^*\) there exists a PPT algorithm \(M^*\), s.t. for every \(x \in L\) the following two random variables are identically distributed:

- \(\text{view}_{V^*}^P(x)\) — the view of \(V^*\) when interacting with \(P\).
- \(M^*(x)\) — the output of \(M^*\).

That is, \(\{\text{view}_{V^*}^P(x)\}_{x \in L} = \{M^*(x)\}_{x \in L}\) as a multiset.
Details

- Too strong a requirement! No non-trivial languages have such proofs.

- Modification: $M^*$ can output $\bot$ with probability $\leq \frac{1}{2}$. If $M^*(x) \neq \bot$ then $\text{view}^{P}_{V^*}(x) = M^*(x)$. (Perfect ZK)

- Alternate modification: $\{\text{view}^{P}_{V^*}(x)\}_{x \in L}$ and $\{M^*(x)\}_{x \in L}$ are statistically close. (Statistical ZK)

- Yet another: $\{\text{view}^{P}_{V^*}(x)\}_{x \in L}$ and $\{M^*(x)\}_{x \in L}$ cannot be distinguished in probabilistic polynomial time.
Intuition

- Perfect ZK: The distributions $\text{view}_{V^*}(x)$ and $M^*(x)$ are same

- Statistical ZK: The distributions $\text{view}_{V^*}(x)$ and $M^*(x)$ are close (so that even an omnipotent adversary cannot make a difference)

- Computational ZK: The distributions $\text{view}_{V^*}(x)$ and $M^*(x)$ cannot be distinguished by a PPT adversary
Complexity classification

The classes of languages that have computational/statistical/perfect zero-knowledge proofs:

\[ \text{BPP} \subset \text{PZK} \subset \text{SZK} \subset \text{CZK} = \text{IP} \]

\[ \text{BPP} \subset \text{PZK} \]: Trivial, uses no interaction: \( \text{PZK} \) can verify by himself whether \( x \in L \).

Reminder: \( \text{BPP} \) — set of problems that can be decided by probabilistic polynomial-time Turing machines.
Example: GI ∈ PZK

$P$ knows an isomorphism $\phi : G_1 \to G_2$.

1. $P$ generates a random permutation $\pi$ of $G_2$-s vertices. She sends $G' \leftarrow \pi(G_2)$ to $V$.

2. $V$ generates a random $\sigma \leftarrow \{0, 1\}$ and sends it to $P$.

3. If $\sigma = 1$, $P$ sets $\tau \leftarrow \pi \circ \phi$, otherwise she sets $\tau \leftarrow \pi$. She sends $\tau$ to $V$.

4. $V$ checks that $\tau(G_{\sigma}) = G'$.

Intuition: $\pi(\phi(G_1)) = \phi(G_2) = G'$. 
\[
\text{NP} \subseteq \text{CZK}
\]

- To show that there are CZK proofs for every NP-language, it is sufficient to show a proof for one concrete NP-complete language.

- A graph \( G \) can be colored with \( c \) colors when there exists a coloring of the vertices of \( G \) with \( c \) colors so that for no edge, the vertices connected to this edge are colored with the same color.

- \( \chi(G) \) - the chromatic number of \( G \). Minimum \( c \) so that \( G \) can be colored with \( c \) colors.

- \( 3COL \): the set of graphs with \( \chi(G) \leq 3 \). This language is NP-complete. Say the colors are R, G, B.
CZK protocol for 3COL

Common input: $G$. $P$ wants to prove that she knows a coloring $C : V(G) \rightarrow \{R, G, B\}$ in CZK. Iterate the next protocol $|E(G)|^2$ times:

- $P$ chooses a random permutation $\pi$ of colors. She encrypts the color $\pi(C(v))$ for every vertex $v$, using a probabilistic public-key cryptosystem, by using a different key for every vertex. $P$ sends to $V$ all ciphertexts together with the correspondence between them and the vertices.

- $V$ chooses a random edge $e = (v, w)$ of the graph, and sends $e$ to $P$.

- $P$ sends the decryption keys $D_v$ and $D_w$ to $V$.

- $V$ computes $\pi(C(v))$ and $\pi(C(w))$ and verifies that they are different.
Correctness of this protocol

- If $P$ knows the corresponding 3-coloring, $V$ will never detect an incorrectly colored edge. Thus, $V$ will accept with probability 1.

- If $\chi(G) > 3$ then $\pi(C(v)) = \pi(C(w))$ in all steps with probability $\geq |E|^{-1}$. After $|E|^2$ steps the probability that $V$ will accept is exponentially small.
Reminder: Honest-Verifier ZK

- A ZK protocol is *honest-verifier*, if it is required to be ZK only in the case when the verifier follows the protocol

- Usually, in the case of HVZK protocols the verifier is only required to send random strings

- Every ZK protocol requires at least four rounds

- HVZK is achievable in 3 rounds
Non-Interactive ZK

- A ZK protocol is noninteractive, if it consists of only one step: prover sending some information to verifier

- A NIZK protocol exists only if $P$ and $V$ have access to some common, publicly available source of random strings (beacon)

- NIZK honest-verifier protocols exist in random-oracle model

- Many other related problems...
ZK and Commitment Schemes

- ZK: done
- Commitment schemes: next
Commitment Schemes

- $P$ has private key $K$. Using this key and a random value $r$, she can commit to some $x$ by sending $C_K(x; r)$ to $V$

- Later, $P$ can reveal $x$ and $V$ can verify that this is the value that was previously committed

- Commitment scheme must be hiding: $V$ will not be able to compute $x$ from its commitment $C_K(x; r)$

- Commitment scheme must be binding: $P$ cannot generate an $x' \neq x$, and an $r'$, s.t. $C_K(x; r) = C_K(x'; r')$
Application: Joint coin tossing

- Alice and Bob want to decide on something by tossing a coin over a phone. How to do this securely?

- Solution: Alice commits to a random bit \( b_A \leftarrow_R \{0, 1\} \), and sends \( C_K(b_A; r) \) to Bob

- Bob selects a random bit \( b_B \leftarrow_R \{0, 1\} \) and sends it to Alice

- Alice decommits \( b_A \)

- Alice and Bob compute the coin toss as \( b_A \oplus b_B \)
**Pedersen commitment**

Assume that $p = 2q + 1$ is a safe prime (i.e., $q$ is also prime)

Set-up Let $h$ be a generator of $G_q$, a subgroup of $\mathbb{Z}_p^*$ of prime order $q$. Let $g \leftarrow_R G$

- Commitment: $C_K(m; r) = g^m h^r \mod p$ where $r \leftarrow_R \mathbb{Z}_q$

- Opening: reveal $m$ and $r$
Proof of security

● Unconditional hiding:

⋆ Since $r$ is a random element of $\mathbb{Z}_q$ then $g^m h^r$ is a random element of $G$, independently of the choice of $m$

● Computational binding:

⋆ Given $(m; r), (m'; r')$, s.t. $g^m h^r = g^{m'} h^{r'}$, $m \neq m'$, one can compute $g \leftarrow h^{(r-r')/(m'-m)}$. (This is valid since $m \neq m'$, $q$ is prime and therefore $(m' - m)^{-1}$ exists.) Therefore, the adversary has computed the DL of $g$ in base $h$

● Note that the proofs are similar to the security proofs of Schnorr’s identification scheme
HVZK: protocols about commitments

Pedersen commitment scheme. Proof that $P$ knows how to open $y = C_K(\mu; \rho)$:

- $P$ generates a random $n$ and a random $s$, and sends $a = C_K(n; s) = g^n h^s$ to $V$

- $V$ generates a random $c \leftarrow \{0, 1\}^t$ and sends $c$ to $P$

- $P$ sends $z = n + c\mu$, $w = s + c\rho$ to $V$

- Verifier checks that $C_K(z; w) \overset{?}{=} ay^c$.

We saw security proofs for such protocols during the last lecture
Notation

- The proof in last slide is called *proof of knowledge*

- Denoted: \( PK(y = C_K(\mu; \rho)) \)

- Greek letters denote variables, knowledge of which is to be proved

- Other letters denote variables that are either in public knowledge or secretly owned by some party

- Another example: \( PK(y = C_K(\mu; \rho) \land \mu \neq 0) \) (proof of knowledge of committed non-zero message \( \mu \))
Why commitments are good for ZK?

• Design a 3-round HVZK protocol between $P$ and $V$: $P$ sends the first and the third steps, $V$ sends a random string on the second step.

• In practice, hard to guarantee that $V$ does not cheat

• Solution:
  
  ★ $V$ selects his response $c$ and commits to it before seeing $P$’s first messages

  ★ $P$ sends then her first message, $V$ opens his commitment, and $P$ sends her second message
Advanced example: Auctions

http://www.tcs.hut.fi/~helger/papers/

- You have a limited number of options: bidding $\mu \in [0, H]$

- You bid by encrypting your bid and sending it to some center

- Goal: seller $S$ should not be able to decrypt your bid; but she should get to know the highest bid

- Solution: Encrypt by using the public key of another center $A$ but send encryption to $S$
Advanced example: Auctions, 2

- Assume $E$ is homomorphic: $E_K(m)E_K(m') = E_K(m + m')$

- Instead of bid $\mu$, encrypt $B^\mu$, where $B$ is the maximum number of bidders

- $S$ multiplies all ciphertexts, obtaining $c \leftarrow E_K(\sum_i B^{\mu_i})$. Due to the choice of $B$, this is equal to $E_K(\sum_j \alpha_j B^j)$, where $\alpha_j$ is the number of bidders who bid $j$

- $S$ sends $c$ to $A$, who decrypts $c$, and obtains all values $\alpha_j$. $A$ calculates the highest bid $X_1 = \max_j (\alpha_j \neq 0)$, and sends it to $S$

- $S$ announces $X_1$ to bidders
Advanced example: Auctions, 3

- Nice protocol, but works only when different parties are honest

- Standard solution: Add a ZK proof that every step was correct
  
  \[
  \star \text{ Used in many cryptographic protocols!}
  \]

- Every bidder proves that it encrypted a valid bid \( B^\mu, \mu \in [0, H] \)

- And: \( A \) proves that \( A \) computed \( X_1 \) correctly
\[
PK(y = E_K(B^\mu; \rho) \land (\mu \in [0, H]))
\]

- Denote \( H_j := \lfloor (H + 2^j)/2^{j+1} \rfloor, j = 0 \ldots \lfloor \log_2 H \rfloor \). Then

\[
\mu \in [0, H] \iff \mu = \sum_{j=0}^{\lfloor \log_2 H \rfloor} \mu_j H_j \quad \text{for some } \mu_j \in \{0, 1\} . \quad (1)
\]

- For example, \( \mu \in [0, 10] \iff \mu = 5\mu_0 + 3\mu_1 + \mu_2 + \mu_3 \) and

\[
\mu \in [0, 9] \iff \mu = 5\mu_0 + 2\mu_1 + \mu_2 + \mu_3 .
\]

- ZK proof idea: show in ZK that you know \( \mu_j \) for which the right side (1) holds ("oblivious binary search")
How to prove that $X_1$ is correct?

- You have

$$y = E_K\left(\sum_j \alpha_j B^j\right).$$

You must show that if $j > X_1$ then $\alpha_j = 0$ and if $j = X_1$ then $\alpha_j > 0$.

- Thus, this is equal to the proof that

$$PK(y = E_K(\mu; \rho) \land \mu = B^{X_1} + \mu_2 \land \mu_2 < B^{X_1+1}).$$
Security properties

If $A$ and $S$ do not cooperate:

- $A$ will not be able to change the highest bid or bidder
- $S$ will not get to know anything about the bids
- $A$ will know the statistics (how many bid $j$) but no individual bids
- System can be strengthened: even cooperating $A$ and $S$ will not be able to change the highest bid or bidder
E-voting

- E-voting: can do analogously. Bidder = voter, bid = vote

- $S$ must get to know $\alpha_j$, so instead of $X_1$ a ZK proof of its correctness
  $A$ will send to her the sum $\sum_j \alpha_j B^j$ (simpler!)

- Problem: Can we trust that $S$ and $A$ do not to cooperate?

- If not, another possibility is to share the trust among a larger number
  of authorities
Next lecture

- Secret sharing: How to guarantee that the secret can be recovered only by privileged sets of users?

- Threshold trust: How to guarantee in general that some system will remain secure if a majority of servers are trustworthy?

- Multi-party computation: Everything can be computed securely by using a secret-sharing approach