Introduction to Cryptography

F. Koeune – O. Pereira

Slides 07

Flipping coins . . . over the Internet

Flipping coins … over the Internet

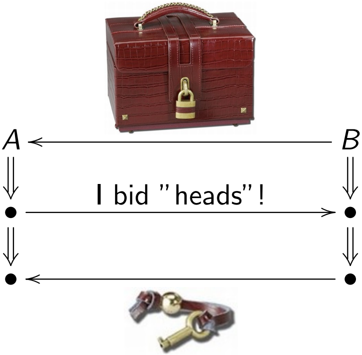
Tentative solution:

$$\begin{tikzcd} A \arrow[r, "\text{I bid ''heads''!}"] \arrow[d, Rightarrow] & B \arrow[d, Rightarrow] \\ \bullet & \bullet \arrow[l, "\text{You lost!}"'] \end{tikzcd}$$

- How can A know whether B is adapting his answer?
- A wants B to :
  - flip a coin independently of her bet,
  - answer honestly

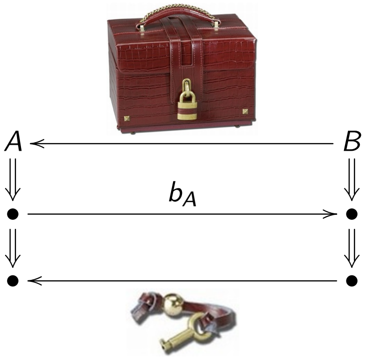
Flipping coins . . . over the Internet

A solution:



1. B sends the outcome of a coin flip to A in a locked box
2. A sends her bet
3. B sends the key of the box

Selecting a random bit



1. B sends a bit b_B to A in a locked box
2. A sends a bit b_A to B
3. B sends the key of the box

   The outcome is $b_A \oplus b_B$

Commitment schemes

What do  and  provide?

1. **Binding** property: once I sent a value locked in the box, I cannot change it anymore
2. **Hiding** property: nobody can tell what is inside the box without the key

Commitment schemes

A triple $\langle \mathrm{Gen}, \mathrm{Com}, \mathrm{Open} \rangle$ of PPT algos:

- Gen probabilistically selects $pk \leftarrow \mathrm{Gen}(1^n)$
  pk is the public key
- Com provides $(c, d) \leftarrow \mathrm{Com}_{pk}(m)$
- Open provides $m := \mathrm{Open}_{pk}(c, d)$
  (or $\perp$ if c and d do not match)

s.t., $\exists$ negl. $\epsilon : \forall n$, $pk \leftarrow \mathrm{Gen}(1^n)$, and $\forall m$:

$$\Pr[\mathrm{Open}_{pk}(\mathrm{Com}_{pk}(m)) \neq m] < \epsilon(n)$$

Assumptions: $|pk| \geq n$ and pk is always generated correctly

Commitment schemes – Hiding

Hiding property:
Given $\Pi := \langle \mathrm{Gen}, \mathrm{Com}, \mathrm{Open} \rangle$ and adversary $\mathcal{A}$, define the experiment $\mathsf{Com}^{\mathsf{hide}}_{\mathcal{A},\Pi}(n)$:

1. Generate $pk \leftarrow \mathrm{Gen}(1^n)$
2. $\mathcal{A}(pk)$ outputs m_0, m_1
3. Choose $b \leftarrow \{0, 1\}$, compute $(c, d) \leftarrow \mathrm{Com}_{pk}(m_b)$, and send c to $\mathcal{A}$
4. $\mathcal{A}$ outputs m'
5. Define $\mathsf{Com}^{\mathsf{hide}}_{\mathcal{A},\Pi}(n) := 1$ iff $m = m'$

Commitment schemes – Hiding

$\Pi := \langle \mathrm{Gen}, \mathrm{Com}, \mathrm{Open} \rangle$ is *perfectly hiding* if $\forall \mathcal{A}$:

$$\Pr[\mathsf{Com}^{\mathsf{hide}}_{\mathcal{A},\Pi}(n)] = \frac{1}{2}$$

$\Pi := \langle \mathrm{Gen}, \mathrm{Com}, \mathrm{Open} \rangle$ is *computationally hiding* if $\forall$ PPT $\mathcal{A}$, $\exists$ negl. ϵ:

$$\Pr[\mathsf{Com}^{\mathsf{hide}}_{\mathcal{A},\Pi}(n)] \leq \frac{1}{2} + \epsilon(n)$$

Commitment schemes – Binding

Binding property:
Given $\Pi := \langle \mathrm{Gen}, \mathrm{Com}, \mathrm{Open} \rangle$ and adversary $\mathcal{A}$, define the experiment $\mathsf{Com}^{\mathsf{bind}}_{\mathcal{A},\Pi}(n)$:

1. Generate $pk \leftarrow \mathrm{Gen}(1^n)$
2. $\mathcal{A}(pk)$ outputs $\langle c, d_0, d_1 \rangle$
3. Define $\mathsf{Com}^{\mathsf{bind}}_{\mathcal{A},\Pi}(n) := 1$ iff:
   a. $\mathrm{Open}_{pk}(c, d_0) = m_0 \neq \perp$
   b. $\mathrm{Open}_{pk}(c, d_1) = m_1 \neq \perp$
   c. $m_0 \neq m_1$

Commitment schemes – Binding

$\Pi := \langle \mathrm{Gen}, \mathrm{Com}, \mathrm{Open} \rangle$ is *perfectly binding* if $\forall \mathcal{A}$:

$$\Pr[\mathsf{Com}^{\mathsf{bind}}_{\mathcal{A},\Pi}(n)] = 0$$

$\Pi := \langle \mathrm{Gen}, \mathrm{Com}, \mathrm{Open} \rangle$ is *computationally binding* if $\forall$ PPT $\mathcal{A}$, $\exists$ negl. ϵ:

$$\Pr[\mathsf{Com}^{\mathsf{bind}}_{\mathcal{A},\Pi}(n)] \leq \epsilon(n)$$

Perfect commitment schemes?

Thm: No commitment scheme is both perfectly binding and hiding.

Intuition:

- If the scheme is perfectly binding, then all the information about the message m is included in c. So, it cannot be perfectly hiding.

Consequence:

- We always need a security parameter...

Perfectly binding commitment schemes

Define $\langle \mathrm{Gen}, \mathrm{Com}, \mathrm{Open} \rangle$:

- $\mathrm{Gen}(1^n)$ sets pk as $(\mathbb{G}, q, g, h)$, where
  - $(\mathbb{G}, q, g)$ is provided by $\mathcal{G}(1^n)$
  - the DDH problem is hard with respect to $\mathcal{G}$
  - h is a random element of $\mathbb{G}$
- $\mathrm{Com}_{pk}(m)$ with $m \in \mathbb{Z}_q$ provides (c, d) where:
  - $c := (g^y, g^m h^y)$ with $y \leftarrow \mathbb{Z}_q$
  - $d := (y, m)$
- $\mathrm{Open}_{pk}(c, d)$ outputs m if it can recompute c from d and pk, or $\perp$ otherwise

Observe:

- Perfectly binding: only one possible "decryption"
- Computationally hiding: breaking DDH reveals m

Perfectly hiding commitment schemes

Attempt:
Define $\langle \mathrm{Gen}, \mathrm{Com}, \mathrm{Open} \rangle$:

- $\mathrm{Gen}(1^n)$ selects pk randomly in $\{0,1\}^n$
- $\mathrm{Com}_{pk}(m)$ provides (c, d) where:
  - $c := m \oplus k$ with $k \leftarrow \{0,1\}^n$
  - $d := (k, m)$
- $\mathrm{Open}_{pk}(c, d)$ outputs m if it can recompute c from d, or $\perp$ otherwise

Observe:

- Perfectly hiding: one-time pad
- Binding???

Perfectly hiding commitment schemes

Define $\langle \mathrm{Gen}, \mathrm{Com}, \mathrm{Open} \rangle$:

- $\mathrm{Gen}(1^n)$ sets pk as $(\mathbb{G}, q, g, h)$, where
  - $(\mathbb{G}, q, g)$ is provided by $\mathcal{G}(1^n)$
  - the DL problem is hard with respect to $\mathcal{G}$
  - h is a random generator of $\mathbb{G}$ ($\log_g(h)$ is unknown)
- $\mathrm{Com}_{pk}(m)$ provides (c, d) where:
  - $c := g^m h^y$ with $y \leftarrow \mathbb{Z}_q$
  - $d := (y, m)$
- $\mathrm{Open}_{pk}(c, d)$ outputs m if it can recompute c from d and pk, or $\perp$ otherwise

Observe:

- Perfectly hiding: h^y is a random element of $\mathbb{G}$
- Computationally binding: If $g^{m_1} h^{y_1} = g^{m_2} h^{y_2}$ and $m_1 \neq m_2$ then $h = g^{(m_1 - m_2)(y_2 - y_1)^{-1}}$.

So,

- We are looking for x so that $h = g^x$
- We have:

$$
\begin{aligned}
& g^{b_1} h^{y_1} = g^{b_2} h^{y_2} \\
\Rightarrow \quad & g^{b_1} g^{x y_1} = g^{b_2} g^{x y_2} \\
\Rightarrow \quad & g^{b_1 + x y_1} = g^{b_2 + x y_2} \\
\Rightarrow \quad & b_1 + x y_1 \equiv b_2 + x y_2 \mod q \\
\Rightarrow \quad & x \equiv (b_2 - b_1)(y_1 - y_2)^{-1} \mod q
\end{aligned}
$$

Quiz

Who should choose the public key?

- In the perfectly binding scheme
    1. The committer?
    2. The receiver?
- In the perfectly hiding scheme
    1. The committer?
    2. The receiver?

Zero-knowledge proofs

The identification problem:

The identification problem

"I am the only one who knows this secret"

How do I prove that?

1. Send the secret?
   No: then the verifier also know my secret. . .
2. Take a private key as secret, and show that I can decrypt a message?
   Could be too much: offers a decryption oracle to the verifier. . .

The identification problem

I want to prove that I am the one who knows this secret, without offering any other knowledge ...

Idea: Make sure that the verifier already knows my answer!

$$
\begin{tikzcd}
P \arrow[d, Rightarrow] & & V \arrow[ll, "Enc_{pk}(m)"'] \arrow[d, Rightarrow] \\
\bullet \arrow[rr, "c"] \arrow[d, Rightarrow] & & \bullet \arrow[d, Rightarrow] \\
\bullet \arrow[d, Rightarrow] & & \bullet \arrow[ll, "m'"'] \arrow[d, Rightarrow] \\
\bullet \arrow[rr, "d"] & & \bullet
\end{tikzcd}
$$

- P proves that he knows the sk matching his public key pk
- $(c, d) \leftarrow \mathrm{Com}(m)$
- d is sent only if $m = m'$

Proofs

"Traditional" mathematical proofs:

> "A list of reasons that shows a statement to be true"

- Non interactive
- No unique verifier in mind

Interactive proofs

Three ingredients:

1. A *prover* P, possibly unbounded
2. A *verifier* V, PPT bounded
3. A language $L \subset \{0,1\}^*$ defining a set of true statements

Motivations:

- Even if P is unbounded, he should not be able to prove wrong things
- V must be able to perform his task effciently
- L can be a lot of things:
  - set of DH tuples $\langle g, g^x, g^y, g^{xy} \rangle$
  - set of pairs of isomorphic graphs
  - set of true theorem statements

Interactive proofs

The pair (P, V) is an *interactive proof system* for L if:

1. **Completeness**: If $x \in L$ then the probability that P does not convince V is negligible in $|x|$
2. **Soundness**: If $x \notin L$ then the probability that any P^* convinces V is negligible in $|x|$

Observations:

- Proofs are probabilistic
- V can be convinced even if P^* is unbounded
- Examples:
  - For the set of DH-tuples: send x
  - For the set of isomorphic graphs: send an isomorphism

Zero-knowledge proofs

Motivation:

- Protect the prover: the verifier should not learn anything but the fact that $x \in L$

Idea:

- Let *trans* be the discussion between P and any PPT V^* on input x.
- It should be feasible to produce something indistinguishable from *trans* just from x

Observations:

- This "simulator" can build *trans* in any order! (So could the verifier if he tries to produce *trans*)
- No verifier can convince that a transcript is "real": he could have produced it himself

Zero-knowledge proofs

(P, V) is a *perfect zero-knowledge* interactive proof system for L if $\forall$ PPT V^*, $\exists$ a PPT simulator $\mathcal{S}_{V^*}$ s.t. $\forall \mathcal{E}$:

$$Pr[\mathcal{E}(trans_{(P,V^*)}(x)) = 1] = Pr[\mathcal{E}(trans_{\mathcal{S}_{V^*}}(x)) = 1]$$

where:

- $trans_{(P,V^*)}(x)$ is the transcript of the interaction of P and V^* on input x
- $trans_{\mathcal{S}_{V^*}}(x)$ is the output of $\mathcal{S}_{V^*}$ on input x
- $\mathcal{E}$ is anyone who tries to distinguish the two transcripts

Remark:

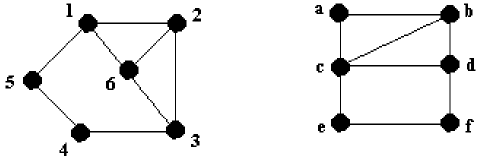
- One could define *computational zero-knowledge*:
  - $\mathcal{E}$ must be PPT
  - the probabilities can have a negligible difference

Graph isomorphism

Two graphs $G := (G_V, G_E)$ and $H := (H_V, H_E)$ are isomorphic if

- $\exists$ a bijection $f : G_V \to H_V$ and
- $(g_1, g_2) \in G_E \Leftrightarrow (f(g_1), f(g_2)) \in H_E$

Are these two graphs isomorphic?



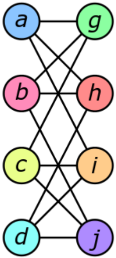
No known algorithm allows deciding in PPT whether two graphs are isomorphic

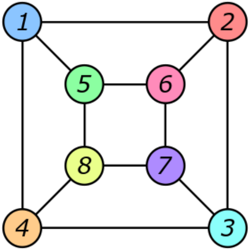
Graph isomorphism

Two graphs $G := (G_V, G_E)$ and $H := (H_V, H_E)$ are isomorphic if

- $\exists$ a bijection $f : G_V \rightarrow H_V$ and
- $(g_1, g_2) \in G_E \Leftrightarrow (f(g_1), f(g_2)) \in H_E$

Example:





Proof of Graph isomorphism

On input $G := (G_V, G_E)$ and $H := (H_V, H_E)$ (isomorphic):

1. P computes (or knows) a bijection $f : G_V \to H_V$
2. P repeats n times:
   a. P publishes a graph (I_V, I_E) built as follows:
      - select a random bijection $g : G_V \to I_V$,
      - build I_E s.t. (G_V, G_E) and (I_V, I_E) are isomorphic
   b. V sends a random bit b to P
   c. P answers with h where:
      - $h := g^{-1}$ if $b = 0$
      - $h := fg^{-1}$ if $b = 1$
3. V accepts the proof if, every time, h witnesses that:
   - (I_V, I_E) is isomorphic to (G_V, G_E) when $b = 0$
   - (I_V, I_E) is isomorphic to (H_V, H_E) when $b = 1$

Proof of Graph isomorphism

Completeness:

- P can answer all challenges

Soundness:

- If G and H are not isomorphic, then P has a probability $\frac{1}{2}$ of not being able to answer the challenge
- That makes a probability 2^{-n} of being able to convince P

Proof of Graph isomorphism

Perfect zero-knowledge: Build $\mathcal{S}_{V^*}$ as follows:

1. Start V^* and feed it with G and H
2. Repeat until $trans_{\mathcal{S}_{V^*}}$ contains n transcripts:
   a. Flip a coin c
   b. Build a graph I, as in the normal proof, but
      - isomorphic to G if $c = 0$
      - isomorphic to H if $c = 1$
   c. Send I to V^* and wait for b
   d. If $c \neq b$ then rewind V^* where it was when entering this iteration and retry
   e. If $c = b$ then compute the permutation h that would be provided in the protocol, and append $\langle I, c, h \rangle$ to $trans_{\mathcal{S}_{V^*}}$
3. Output $trans_{\mathcal{S}_{V^*}}$

Proof of Graph isomorphism

Observations:

- $\mathcal{S}_{V^*}$ tries to guess b, and restart/reboot V^* when he fails
- Failure probability is $\frac{1}{2}$ each time
- If $\mathcal{S}_{V^*}$ makes up to n^2 attempts, he wins excepted with negligible probability
- The simulated transcript is distributed as the real one

Σ-*protocols*

A family of:

- efficient,
- 3-moves,
- honest-verifier

zero-knowledge protocols.

Schnorr's protocol [1988]

Let $\mathbb{G}$ be a group of prime order q with generator g

$$\begin{tikzcd}
P \arrow[rrr, "g^r"] \arrow[d, Rightarrow] & & & V \arrow[d, Rightarrow] \\
\bullet \arrow[d, Rightarrow] & & & \bullet \arrow[lll, "e"'] \arrow[d, Rightarrow] \\
\bullet \arrow[rrr, "r + e \cdot x \bmod q"] & & & \bullet
\end{tikzcd}$$

P proves knowledge of x to V who has g^x

1. P chooses $r \leftarrow \mathbb{Z}_q$ and commits through g^r
2. V challenges with a random $e \leftarrow \mathbb{Z}_{2^n}$
3. P responds with $f := r + e \cdot x \bmod q$
4. V accepts if $g^f = g^r \cdot (g^x)^c$

Schnorr's protocol

$$\begin{tikzcd}
P \arrow[rr, "g^r"] \arrow[d, Rightarrow] & & V \arrow[d, Rightarrow] \\
\bullet \arrow[d, Rightarrow] & & \bullet \arrow[ll, "e"'] \arrow[d, Rightarrow] \\
\bullet \arrow[rr, "f := r + e \cdot x \bmod q"] & & \bullet
\end{tikzcd}$$

Completeness: obvious

Soundness:

- In order to reply with non-negligible probability, P must be able to respond to more than 2 challenges, say e and e'
- Then $g^f/(g^x)^e = g^{f'}/(g^x)^{e'}$ and $x = \frac{f-f'}{e-e'}$

Honest verifier zero-knowledge:

- Choose e, f at random and compute $g^r := g^f/(g^x)^e$
  (This does not works if, say, V computes $e := \mathcal{H}^s(g^r)$)

Σ-*protocols*

Π is a Σ-protocol for relation R if:

- It is a 3-move protocol with completeness, made of a *commitment*, followed by a random *challenge*, and ending with a *response*
- For any pair (a, e, f) and (a, e', f') of accepting conversations on input x where $e \neq e'$, one can efficiently compute $w : (x, w) \in R$
- There is an efficient simulator that, on input x, e, produces (a, f) such that (a, e, f) is distributed as in a normal proof.

Not just proof that $x \in L = \{x : \exists w \text{ s.t. } (x, w) \in R\}$, but **proof of knowledge** of a witness $w : (x, w) \in R$.

Non-interactive ZK

Honest verifier ZK can be useful!

Let $\mathcal{H}$ be a random function:

- Compute $e := \mathcal{H}(a, x)$ and send *non-interactive* proof (a, e, f)!

Intuition:

- $\mathcal{H}$ makes sure that you pick a and x before seeing e: the only way of seeing the right e is to evaluate $\mathcal{H}$ on a, x!

Challenge:

- We cannot use a random function! (Too big.)
- We cannot use a PRF: the prover needs to evaluate it, so he needs k, which makes it not random

The Random Oracle Model (ROM)

Solution:

- make "as if" a random function were available as an oracle for everyone
- assume that, in reality, it will work to replace it with a good hash function

Is it a sound methodology?   Of course not!

- A real hash function has all its I/Os defined in advance
  A RO has its I/Os unknown as long as it has not been queried
- A real hash function might be computed in different ways
  The output of a RO can only be known by querying the RO
- . . .

The Random Oracle Model

Why do we use it, then?

- Many standardized and convincing schemes can only be proven using the ROM (or a similar model)
- In practice, good hash functions give a random-looking output for any fresh input (of a given length)

What do we have, then?

- A "not-too-bad" methodology for arguing that a scheme is secure
- Something that is believed to be better than no proof at all

Non-interactive ZK in the ROM

Make a Σ-protocol (a, e, f) non interactive:

- Compute $e := \mathcal{H}(a, x)$ and send *non-interactive* proof (a, e, f)!

The resulting protocol is ZK in the ROM.
The NI simulator $\mathcal{S}$:

- needs to produce *trans* distributed as in a real execution
- can control the RO $\mathcal{H}$ that $\mathcal{E}$ uses

Strategy for $\mathcal{S}$ to build $trans = (a, e, f)$

1. pick random e,
2. run the Σ-protocol simulator, and get (a, e, f)
3. decide that $\mathcal{H}(a, x) = e$

This works since: e is uniformly random, the simulator works, and a is a fresh value, unlikely to have been queried to $\mathcal{H}$ before.

Non-interactive ZK in the ROM

Make a Σ-protocol (a, e, f) non interactive:

- Compute $e := \mathcal{H}(a, x)$ and send *non-interactive* proof (a, e, f)!

The resulting protocol is sound in the ROM. Sketch:

- Let P^* output valid (a, e, f) with non negl. proba
- P^* must have made a (a, x) query to $\mathcal{H}$ otherwise, unlikely to have $\mathcal{H}(a, x) = e$
- Since P^* outputs (a, e, f), he must be able to do so on a non-negligible number of outputs of $\mathcal{H}(a, x)$
- So:
  1. Start P^*, play the $\mathcal{H}$ honestly, get (a, e, f)
  2. Restart P^*, answer the $\mathcal{H}$ in the same way until (a, x) query sent to $\mathcal{H}$, answer that one with rand. $e' \neq e$
  3. With non negligible probability, receive (a, e', f')

Non-interactive ZK in the ROM

Conclusion:

- We can transform any HVZK Σ-protocol into a NIZK protocol
- But security only holds in the ROM

This is the basis of most signature schemes used today
(Stay tuned...)

Proving statements about ElGamal ciphertexts

ElGamal encryption in prime-order group:

- Public key: $(g, h) := (g, g^x)$
- Ciphertext: $(c_1, c_2) := (g^y, m \cdot g^{xy})$

Statement:

- (c_1, c_2) is an encryption of m under (g, h)
- witness: either x or y

Reformulation:

L contains all (g_1, g_2, g_3, g_4) s.t. $\log_{g_1}(g_2) = \log_{g_3}(g_4)$

- Either $(g_1, g_2, g_3, g_4) := (g, g^x, g^y, g^{xy})$ (witness is x)
- Or $(g_1, g_2, g_3, g_4) := (g, g^y, g^x, g^{xy})$ (witness is y)

Chaum-Pedersen protocol

Let $\mathbb{G}$ be a group of prime order q with generator g

$$\begin{tikzcd}
P \arrow[r, "g_1^r , g_3^r"] \arrow[d, Rightarrow] & V \arrow[d, Rightarrow] \\
\bullet \arrow[d, Rightarrow] & \bullet \arrow[l, "e"'] \arrow[d, Rightarrow] \\
\bullet \arrow[r, "r + e \cdot x \bmod q"] & \bullet
\end{tikzcd}$$

P proves that $\log_{g_1}(g_2) = \log_{g_3}(g_4)(= x)$

1. P chooses $r \leftarrow \mathbb{Z}_q$ and commits through g_1^r, g_3^r
2. V challenges with a random $e \leftarrow \mathbb{Z}_{2^n}$
3. P responds with $f := r + e \cdot x \bmod q$
4. V accepts if $g_1^f = g_1^r \cdot (g_2)^e$ and $g_3^f = g_3^r \cdot (g_4)^e$

Chaum-Pedersen protocol

Let $\mathbb{G}$ be a group of prime order q with generator g

$$\begin{tikzcd}
P \arrow[rr, "g_1^r, g_3^r"] \arrow[d, Rightarrow] & & V \arrow[d, Rightarrow] \\
\bullet \arrow[d, Rightarrow] & & \bullet \arrow[ll, "e"'] \arrow[d, Rightarrow] \\
\bullet \arrow[rr, "r + e \cdot x \bmod q"] & & \bullet
\end{tikzcd}$$

Completeness: obvious

Soundness:

- If P can prove with $((a_1, a_3), e, f)$ and $((a_1, a_3), e', f')$ then $\log_{g_1}(g_2) = \log_{g_3}(g_4) = \frac{f-f'}{e-e'}$

Honest verifier zero-knowledge:

- Choose e, f at random and compute $g_1^r := g_1^f/(g_2)^e$ and $g_3^r := g_3^f/(g_4)^e$

Proving OR statements

Suppose we have:

- a Σ-protocol Π_0 for proving that $x_0 \in L_0$
- a Σ-protocol Π_1 for proving that $x_1 \in L_1$

Combining proofs:

- Proving that $x_0 \in L_0 \wedge x_1 \in L_1$ is trivial
- Can we prove that $x_0 \in L_0 \vee x_1 \in L_1$?

Applications:

- I know one of the DL of $(h_1, \ldots, h_n)$ in base g (anonymous authentication)
- This is an encryption of 0 or 1 (election)

Disjunctive proofs [CDS94]

Suppose prover has $w_i : (x_i, w_i) \in R_i$ (but not w_{1-i})

1. P selects random e_{1-i} and runs $\mathcal{S}_{1-i}$ to get a proof $(a_{1-i}, e_{1-i}, f_{1-i})$
2. P selects a_i as from Π_i's definition
3. P commits on (a_0, a_1) to V
4. V challenges with e
5. P computes $e_i = e - e_{1-i} \bmod 2^n$ and f_i from (w_i, a_i, e_i)
6. V accepts if (a_0, e_0, f_0) and (a_1, e_1, f_1) check for Π_0 and Π_1 and $e_0 + e_1 = e \bmod 2^n$

Disjunctive proofs

$$\begin{tikzcd}
P \arrow[rrr, "a_0, a_1"] \arrow[d, Rightarrow] & & & V \arrow[d, Rightarrow] \\
\bullet \arrow[d, Rightarrow] & & & \bullet \arrow[lll, "e"'] \arrow[d, Rightarrow] \\
\bullet \arrow[rrr, "e_0, f_0, e_1, f_1"] & & & \bullet
\end{tikzcd}$$

Completeness: obvious

Soundness:

- P^* has to follow either Π_0 or Π_1

Honest verifier zero-knowledge:

- Choose (e_0, e_1) at random, run $\mathcal{S}_0$, $\mathcal{S}_1$ to get (a_0, e_0, f_0) and (a_1, e_1, f_1)
- Simulated transcript is $(a_0, a_1, e_0 + e_1 \bmod 2^n, e_0, f_0, e_1, f_1)$

Conclusions

Zero-knowledge proof systems

- I convince you that this statement is true
- This is the only thing you learn
- You cannot use my proof to convince anyone else (interactive case)

References (see Moodle or search online):

- Ivan Damgård and Jesper Buus Nielsen: Commitment Schemes and Zero-Knowledge Protocols
- Ivan Damgård: On Σ-protocols