Security protocols

Lecture 9

Commitment schemes

Coin tossing over the phone

Alice and Bob want to toss coins over telephone to decide who gets something they both want.

Protocol 0

- Bob phones Alice and says he'll flip a coin
- Alice guesses "Tails"
- Bob (honest) says "Sorry, it was Heads"

Coin tossing over the phone

Alice and Bob want to toss coins over telephone to decide who gets something they both want.

Protocol 0

- Bob phones Alice and says he'll flip a coin
- Alice guesses "Tails"
- Bob (honest) says "Sorry, it was Heads"

Protocol 1

- Bob sends Alice messages "head" and "tail" encrypted by a one-way function f
- Alice guesses which one of them is an encryption of the head
- Bob tells Alice whether his guess was correct
- If Alice does not believe him, Bob sends f to Alice

Coin tossing over the phone

Protocol 2

- Alice chooses p and q prime (congruent to 3 mod 4) and sends Bob n = pq
- Bob chooses a random x ∈ {1, …, n/2} and sends Alice $y = x^2 \bmod n$ (if Alice guess x correctly, she wins)
- Alice computes the four roots (easy if you know p and q, hard if not), at random chooses v as one of the two roots < n/2 and sends v to Bob
- If $v = \pm x \bmod n$, Bob sends "OK, you win" to Alice.
  If $v \neq \pm x$, Bob sends "I win" and x to Alice, to prove he knew it (he can now factor n; Alice should check that $x^2 = y \bmod n$)

Coin tossing over the phone

Basic requirements: In any good coin tossing protocol both parties should influence the outcome and should accept the outcome. In addition, both outcomes should have the same probability.

Generalized requirements: for a coin tossing protocol:

- The outcome of the protocol is an element from the set {0, 1, reject}.
- If both parties behave correctly, the outcome should be from the set {0, 1}.
- If it is not the case that both parties behave correctly, the outcome should be reject.

Coin tossing over the phone

Problem: In some coin tossing protocols one party can find out the outcome sooner than the second party. In such a case if she is not happy with the outcome she can disrupt the protocol - to produce reject or to say "I do not continue in performing the protocol".

A way out is to require that in case of correct behavior no outcome should have probability > 1/2

Coin tossing using a one-way function

- Bob chooses a one-way function f and informs Alice about the definition domain of f
- Alice chooses randomly r1, r2 from domain and sends them to Bob.
- Bob sends to Alice one of the values f(r1) or f(r2).
- Bob announces Alice his guess which of the two values he received.
- Alice announces Bob whether his guess was correct (0) or not (1).
- If Bob wants to verify correctness, Alice has to send to Bob the specification of f.

Coin tossing using a one-way function

- Bob chooses a one-way function f and informs Alice about the definition domain of f
- Alice chooses randomly r1, r2 from domain and sends them to Bob.
- Bob sends to Alice one of the values f(r1) or f(r2).
- Bob announces Alice his guess which of the two values he received.
- Alice announces Bob whether his guess was correct (0) or not (1).
- If Bob wants to verify correctness, Alice has to send to Bob the specification of f.

The protocol is computationally secure. Indeed, to cheat, Alice should be able to find, for randomly chosen r1, r2, such one-way function f that f(r1) = f(r2).

Bit commitment

Alice wants to commit to a value of a bit b, such as the outcome of a Hockey game the next weekend. Her intent is to show Bob she can do this, but she does not want to tell Bob how, or what the result is (so that he won't reduce Alice's winnings).
But nonetheless she wants to give Bob something, that he can use to verify that she knew, after the game has been played.

Bit commitment is used to convince someone that you have certain knowledge, without giving the knowledge to this someone — uses one-way functions

Bit commitment

Alice wants to commit to a value of a bit b, such as the outcome of a Hockey game the next weekend. Her intent is to show Bob she can do this, but she does not want to tell Bob how, or what the result is (so that he won't reduce Alice's winnings).
But nonetheless she wants to give Bob something, that he can use to verify that she knew, after the game has been played.

Bit commitment is used to convince someone that you have certain knowledge, without giving the knowledge to this someone — uses one-way functions

Alice will base her commitment scheme on discrete logs, so sets a prime p and primitive root q and makes them public. She now chooses a random x whose second bit is b, and sends $y = g^x \bmod p$ to Bob. It is easy to compute discrete log mod 2 but hard to compute discrete log mod 4. When Alice reveals the value x to Bob, he can check that y and $g^x \bmod p$ are the same.

Commitment schemes

Two-phase protocol between Sender and Receiver

1. **commit**: Sender produces a commitment of some value m for Receiver

2. **reveal**: Sender opens the commitment for Receiver. Receiver can verify the correctness of the commitment

- **hiding** - Receiver cannot compute anything about m from its commitment (computational vs. perfect hiding)
- **binding** - Sender cannot open the commitment as a different value m’ ≠ m (computational vs. perfect binding)

Bit Commitment Scheme

Let f : {0, 1} x X --> Y . A commitment to a b ϵ {0, 1}, or an encryption of b, is any value (called a **blow**) f(b, x) where x ϵ X

- The sender sends a bit b he wants to commit to, in an encrypted form, to the receiver.
- If required, the sender sends to the receiver additional information that enables the receiver to get b.

Hiding (privacy): For no b ϵ {0, 1} and no x ϵ X, it is feasible for Bob to determine b from B = f(b, x).

Binding: Alice can "open" her commitment b, by revealing (opening) x and b such that B = f(b, x), but she should not be able to open a commitment (blow) B as both 0 and 1.

Correctness: If both follow the protocol, then the receiver will always learn (recover) the committed value b.

Bit Commitment with One-way functions

Commitment phase:

- Alice and Bob choose a one-way function f
- Bob sends a randomly chosen r1 to Alice
- Alice chooses random r2 and her committed bit b and sends to Bob f (r1, r2, b).

Revealing phase:

- Alice sends to Bob r2 and b
- Bob computes f (r1, r2, b) and compares with the value he has already received.

Bit Commitment with Hash functions

Commitment phase:

- To commit to a w choose a random r and make public h(w,r).

Revealing phase:

- reveal r and w.

For this application the hash function h has to be one-way: from h(w,r) it should be unfeasible to determine w or r.

BC based on Goldwasser-Micali scheme

Let p, q be large primes, ($p \bmod 4 = q \bmod 4 = 3$), $n = pq$,
$m \in QNR(n)$ {quadratic non-residue; $\forall r: r^2 \bmod n \neq m$}

Commitment: $f(b, x) = m^b x^2 \bmod n$ for a random x from X.

Since computation of quadratic residues is in general unfeasible, this bit commitment scheme is hiding.

Since $m \in QNR(n)$, there are no x_1 , x_2 such that $mx_1^2 = x_2^2 \bmod n$ and therefore the scheme is binding.

Coin Tossing from Bit Commitment

- Alice tosses a coin, and commits itself to its outcome b_A (say to 0 (1) if the outcome is head (tail)) and sends the commitment to Bob.
- Bob also tosses a coin and sends the outcome b_B to Alice.
- Alice opens her commitment to Bob (so he starts to know b_A)
- Both Alice and Bob compute $b = b_A \oplus b_B$.

Observe that if at least one of the parties follows the protocol, that is it tosses a random coin, the outcome is indeed a random bit.

After step 2 Alice will know what the outcome is, but Bob does not. So Alice can disrupt the protocol if the outcome is to be not good for her.

A Commitment scheme based on Discrete Log

Alice wants to commit herself to an $m \in \{0, \ldots, q-1\}$.

- Bob randomly chooses primes p and q such that $q*k + 1 = p$
- Bob chooses random generators g and h of the subgroup G of order q. Bob sends p, q, g and h to Alice.

Commitment phase:

- To commit to an $m \in \{0, \ldots, q-1\}$, Alice chooses a random $r < q$, and sends $c = g^r h^m \bmod p$ to Bob.

Revealing phase:

- Alice sends r and m to Bob who then verifies whether $c = g^r h^m \bmod p$.

Bit Commitment using Encryption

Commit phase:

- Bob generates a random string r and sends it to Alice
- Alice commit herself to a bit b using a key k through an encryption $E_k(r, b)$ and sends it to Bob.

Revealing phase:

- Alice sends the key k to Bob.
- Bob decrypts the message to learn b and to verify r.

Comment: without Bob's random string r Alice could find a different key q such that $E_k(b) = E_q(\neg b)$.

Electronic voting

Let com(r, m) = $g^r h^m$ mod p denote commitment to m in the commitment scheme based on discrete logarithm.
If $r_1, r_2, m_1, m_2 \in Z_n$, then

$$com(r_1, m_1) * com(r_2, m_2) = com(r_1 + r_2, m_1 + m_2).$$

Commitment schemes with such a property are called **homomorphic commitment schemes**.

Homomorphic schemes can be used to cast yes-no votes of n voters $V_1, \ldots, V_n$, by the trusted authority T for whom E and D are ElGamal encryption and decryption algorithms (with multiplicative homomorphic property).

Electronic voting

Each voter V_i chooses his vote $m_i \in \{0, 1\}$, a random $r_i \in \{0, \ldots, q-1\}$ and computes his voting commitment $c_i = com(r_i, m_i)$.

Then V_i makes c_i public and sends $E(g^{r_i})$ to T.

T computes $D(\prod_{i=1}^{n} E(g^{r_i})) = \prod_{i=1}^{n} g^{r_i} = g^r$ where $r = \sum_{i=1}^{n} r_i$ and makes public g^r.

Now, anybody can compute the result s of voting from publicly known c_i and g^r since $h^s = g^{-r} . \prod_{i=1}^{n} c_i$ with $s = \sum_{i=1}^{n} m_i$.

s can now be derived from h^s by computing $h^1, h^2, h^3, \ldots, h^n$ and comparing with h^s if the number of voters is not too large.

(all computation modulo p)

Oblivious transfer protocol

Design a protocol for sending a message from Alice to Bob in such a way that Bob receives the message with probability 1/2 and "garbage“ with the probability 1/2. Moreover, Bob knows whether he got the message or garbage, but Alice has no idea which one he got.

- Alice chooses two large primes p and q and sends n = pq to Bob.
- Bob chooses a random number x and sends $y = x^2 \bmod n$ to Alice.
- Alice computes four square roots $\pm x_1, \pm x_2$ of y (mod n) and sends one to Bob (she can do it, but has no idea which of them is x)
- Bob checks whether the number he got is congruent to x. If yes, he has received no new information. Otherwise, Bob has two different square roots modulo n and can factor n. Alice has no way of knowing whether this is the case.

1-out-of-2 oblivious transfer problem

Alice sends two messages to Bob in such a way that Bob can choose which of the messages he receives (but he cannot choose both), but Alice cannot learn Bob's decision.

Can be imagine as a box with three inputs – Alice inputs x_0 and x_1 and Bob inputs a bit i – and Bob gets as the output x_i

A generalization of 1-out-of-2 oblivious transfer problem is two-party oblivious circuit evaluation problem:

Alice has a secret i and Bob has a secret j and they both know some function f. At the end of protocol the following conditions should hold:

- Bob knows the value f(i, j), but he does not learn anything about i.
- Alice learns nothing about j and nothing about f(i, j).

Implementation of oblivious transfer protocols

- Alice generates two key pairs for a PKC P and sends both her public keys p_1, p_2 to Bob.
- Bob chooses a random secret key k for a SKC S, encrypts it by one of Alice's public keys, p_1 or p_2, and sends the outcome to Alice.
- Alice uses her two secret keys to decrypt the message she received. One of the outcomes is garbage g, another one is k, but she does not know which one is k.
- Alice encrypts her two secret messages, one with k, another with g and sends them to Bob.
- Bob uses S with k to decrypt both messages he got and one of the attempts is successful. Alice has no idea which one.

BIT COMMITMENT from 1-out-2 oblivious transfer

Using 1-out-of-2 oblivious transfer box (OT-box) one can design a bit commitment scheme as follows:

Commitment phase:

- Alice selects a random bit r and her commitment bit b and inputs $x_0 = r$ and $x_1 = r \oplus b$ into the OT-box.
- Alice sends a message to Bob telling him that his turn follows.
- Bob selects a random bit c, inputs c into the OT-box and records the output x_c .

Revealing phase:

- Alice sends r and b to Bob.
- Bob checks to see if $x_c = r \oplus (b \wedge c)$

1-out-of-n oblivious transfer

Mental Poker by phone

Basic requirements (for playing poker with 52 cards):

- Initial hands (sets of 5 cards) of both players are equally likely.
- The initial hands of Alice and Bob are disjoint.
- Both players always know their own hands but not that of the opponent.
- Each player can detect, eventually, cheating of the other player.

A commutative cryptosystem is used with all functions kept secret.

Mental Poker by phone

Players agree on some numbers w_1, …, w_{52} as the names of cards.

- Bob encrypts cards and tells $E_B(w_1)$, …, $E_B(w_{52})$, in a randomly chosen order, to Alice.
- Alice chooses five of the items $E_B(w_i)$ as Bob's hand and tells them Bob. Bob uses D_B to decrypt his hand.
- Alice chooses another five of $E_B(w_i)$, encrypts them with E_A and sends them to Bob.
- Bob applies D_B to all five values $E_A(E_B(w_i))$ he got from Alice and sends $E_A(w_i)$ to Alice as Alice's hand. At this point both players have their hands and poker can start.

Mental Poker by phone with three players

- Alice encrypts 52 cards w_1, …, w_{52} with E_A and sends encryptions, in a random order, to Bob.
- Bob, who cannot decode the encryptions, chooses 5 of them, randomly. He encrypts them with E_B and sends $E_B(E_A(w_i))$ to Alice and the remaining 47 encryptions $E_A(w_i)$ to Carol.
- Carol, who cannot decode any of the encryptions, chooses five of them randomly, encrypts them also with her key and sends Alice $E_C(E_A(w_i))$.
- Alice, who cannot read encrypted messages from Bob and Carol, decrypt them with her key and sends back to the senders, five $D_A(E_B(E_A(w_i))) = E_B(w_i)$ to Bob, five $D_A(E_C(E_A(w_i))) = E_C(w_i)$ to Carol.
- Bob and Carol decrypt encryptions they got to learn their hands.

Mental Poker by phone with three players

- Bob and Carol decrypt encryptions they got to learn their hands.
- Carol chooses randomly 5 other messages $E_A(w_i)$ from the remaining 42 and sends them to Alice.
- Alice decrypt messages to learn her hand.

Additional cards can be dealt with in a similar manner. If either Bob or Carol wants a card, they take an encrypted message $E_A(w_i)$ and go through the protocol with Alice. If Alice wants a card, whoever currently has the deck sends her a card.

Age difference finding (Millionaire problem)

Alice and Bob want to find out who of them is older without disclosing any other information about their age. Let age of Bob be j; and age of Alice be i. (neither Bob nor Alice are older than 100 years).

- Bob chooses a random $x \in \{1, .., 100\}$, computes $k = E_A(x)$ and sends to Alice $s = k - j$.
- Alice first computes the numbers $y_u = D_A(s + u)$; $1 \leq u \leq 100$, then chooses a large random prime p and computes numbers $z_u = y_u \mod p$; $1 \leq u \leq 100$, and verifies that for all $u \neq v$: $|z_u - z_v| \geq 2$ (if this is not the case, Alice choose a new p)
- Finally, Alice sends Bob the following sequence (order is important) $z_1, \ldots, z_i, z_{i+1} + 1, \ldots, z_{100} + 1, p$
- Bob checks whether j-th number in the above sequence is congruent to x modulo p. If yes, Bob knows that $i \geq j$, otherwise $i < j$.

Zero-knowledge proofs/protocols

A zero-knowledge proof or protocol is an interactive process by which one party (the Prover – often called Peggy) can convince another party (the Verifier – called Victor) that a particular statement is true, without conveying any additional information apart from the fact that the statement is indeed true.

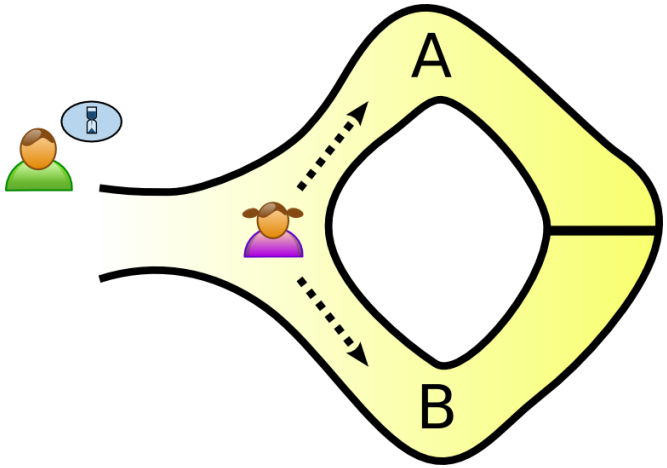
Zero-knowledge proof protocols are a special type of so-called interactive proof systems.

The Ali Baba cave

Quisquater (1990):
How to Explain Zero-Knowledge Protocols to Your Children

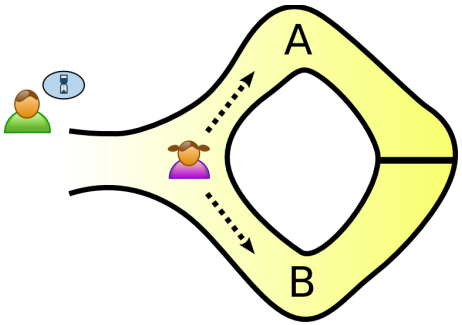
A cave with a magic door opening on a secret word.

Peggy knows a secret word opening the door in cave.
How can she convince Victor about it without revealing this secret word?

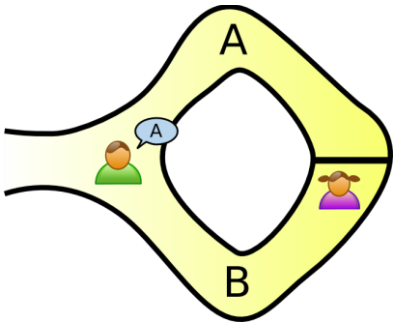


The Ali Baba cave

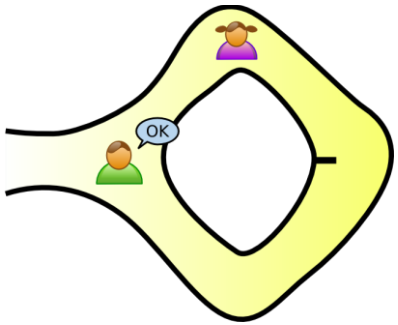
- (A cave with a magic door opening on a secret word)
- Peggy knows a secret word opening the door in cave. How can she convince Victor about it without revealing this secret word?



Peggy randomly takes either path A or B, while Victor waits outside



Victor chooses an exit



Peggy reliably appears at the exit Victor names

Interactive proof protocols

For example, both Prover and Verifier posses an input x and Prover wants to convince Verifier that x has a certain Property and that Prover knows how to prove that.

The interactive proof system consists of several rounds. In each round Prover and Verifier alternatively do the following.

- Receive a message from the other party.
- Perform a (private) computation.
- Send a message to the other party.

Communication starts usually by a challenge of Verifier and a response of Prover. At the end, Verifier either accepts or rejects Prover's attempts to convince Verifier.

Interactive proof protocols

An interactive proof protocol is said to be an interactive proof system for a secret/knowledge or a decision problem P if the following properties are satisfied provided that Prover and Verifier posses an input x (or Prover has secret knowledge) and Prover wants to convince Verifier that x has certain properties and that Prover knows how to prove that (or that Prover knows the secret).

(Knowledge) Completeness: If x is a yes-instance of P, or Peggy knows the secret, then Vic always accepts Peggy's "proof" for sure.

(Knowledge) Soundness: If x is a no-instance of P, or Peggy does not know the secret, then Vic accepts Peggy's "proof" only with very small probability.

Graph non-isomorphism

A simple interactive proof protocol exists for a computationally very hard graph non-isomorphism problem.

Input: Two graphs G_1 and G_2 , with the set of nodes {1, …, n}

Protocol: Repeat m times the following steps:

- Victor chooses randomly an integer i ϵ {1, 2} and a permutation π of {1, …, n}. Victor then computes the image H of G_i under the permutation π and sends H to Peggy.
- Peggy determines the value j such that G_j is isomorphic to H, and sends j to Victor.
- Victor checks to see if i = j.

Victor accepts Peggy's proof if i = j in each of m rounds.

Graph non-isomorphism

Completeness: If G_1 is not isomorphic to G_2 , then probability that Victor accepts is 1 because Peggy will have no problem to answer correctly.

Soundness: If G_1 is isomorphic to G_2 , then Peggy can deceive Victor if and only if she correctly guesses m times those i's Vic chooses randomly. The probability that this can happen is 2^{-m}.

Observe that Vic's computations can be performed in polynomial time (with respect to the size of graphs).

Zero-knowledge proofs

An interactive proof systems has the property of being zero-knowledge if the Verifier, that interacts with the honest Prover of the system, learns nothing from their interaction beyond the validity of the statement being proved.

Different variants of zero-knowledge proof systems concern the strength of this distinguishability. In particular, perfect or statistical zero-knowledge refer to the situation where the simulator's output and the Verifier's output are indistinguishable in an information theoretic sense.

Computational zero-knowledge refer to the case there is no polynomial time distinguishability.

More formally

A zero-knowledge proof of a theorem T is an interactive two party protocol, in which Prover is able to convince Verifier who follows the same protocol, by the overwhelming statistical evidence, that T is true, if T is indeed true, but no Prover is able to convince Verifier that T is true, if this is not so.

In addition, during interactions, Prover does not reveal to Verifier any other information, except whether T is true or not. Consequently, whatever Verifier can do after he gets convinced, he can do just believing that T is true.

Similar arguments hold for the case Prover possesses a secret.

Hamiltonian cycle

Peggy and Victor know a graph G. Peggy will prove to Victor that G has a Hamiltonian cycle (and that she knows how to draw Hamiltonian cycle in G). To do that they perform several times the following rounds:

- Peggy creates randomly a graph H isomorphic to G and commits herself to H before Victor.
- Victor asks Peggy to do randomly one of the following tasks:
  - Show isomorphism between G and H.
    Peggy opens her commitment to make H public and show the required isomorphism.
  - Draw Hamiltonian cycle in H.
    Peggy opens her commitment and shows the Hamiltonian cycle. She can do that because she knows how to do that for G and knows isomorphism between G and H.

Hamiltonian cycle

Completeness: In case Peggy knows how to draw Hamiltonian cycle for G she always does well what Vic asks.

Soundness: In case G does not have Hamiltonian cycle, Peggy is able to do what Vic asks only with some small probability.

Zero-knowledge: None of the responses of Peggy reveals a Hamiltonian cycle of G. It reveals only either isomorphism or a Hamiltonian cycle of H. Victor would need to know both simultaneously in order to be able to draw a Hamiltonian cycle of G.

Zero-knowledge proofs in cryptographic protocols

A cryptographic protocol can be seen as a set of interactive programs to be executed by non-trusting parties. Each party keeps secret her local input.

The protocol specifies the actions parties should take, depending on their local secrets and previous messages exchanged.

The main problem in this setting is how can a party verify that the other parties have really followed the protocol?

The way out: a party A can convince a party B that the transmitted message was the correct one according to the protocol without revealing its secrets.

Zero-knowledge proofs in cryptographic protocols

An idea how to design a reliable protocol

- Design a protocol under the assumption that all parties follow the protocol.
- Transform protocol, using a method to make zero-knowledge proofs out of normal ones, into a protocol with communication based on zero-knowledge proofs, which preserves both correctness and privacy and works even if some parties have an adversary behavior.

Zero-knowledge example

Peggy claims to know a square root s of t mod n = pq, where p and q are large primes (efficiently solving for s is equivalent to efficiently factoring n)

- Peggy uses a random number r , computes $x = r^2 \bmod n$ and sends x to Victor
- Victor asks either for the square root of x, or the square root of xt
- If Victor asked for the square root of x, Peggy lets y = r .
  If he asked for the square root of xt, Peggy lets y = rs .
  She now returns y to Victor
- Victor checks that $y^2 \equiv x$ or $y^2 \equiv xt \pmod n$ depending on what he asked for

If Peggy can do this enough times (she must use different random numbers r each time), Victor will conclude that Peggy knows s, the square root of t

Zero-knowledge example

Peggy claims to know a square root s of t mod n = pq, where p and q are large primes (efficiently solving for s is equivalent to efficiently factoring n)

- Peggy uses a random number r , computes $x = r^2 \bmod n$ and sends x to Victor
- Victor asks either for the square root of x, or the square root of xt
- If Victor asked for the square root of x, Peggy lets y = r .
  If he asked for the square root of xt, Peggy lets y = rs .
  She now returns y to Victor
- Victor checks that $y^2 \equiv x$ or $y^2 \equiv xt \pmod n$ depending on what he asked for

If Peggy can do this enough times (she must use different random numbers r each time), Victor will conclude that Peggy knows s, the square root of t

Zero-knowledge example (Peggy cheating)

Peggy claims to know a square root s of t mod n = pq, where p and q are large primes (efficiently solving for s is equivalent to efficiently factoring n)

- Peggy uses a random number r , computes $x = r^2 \bmod n$ and sends x to Victor
- Victor asks either for the square root of x, or the square root of xt
- If Victor asked for the square root of x, Peggy lets y = r .
  If he asked for the square root of xt, Peggy doesn't know s and must guess y = rs . She now returns y to Victor
- Victor checks that $y^2 \equiv x$ or $y^2 \equiv xt \pmod n$ depending on what he asked for

If Peggy can do this enough times (she must use different random numbers r each time), Victor will conclude that Peggy knows s, the square root of t

Zero-knowledge example (Peggy cheating)

Peggy claims to know a square root s of t mod n = pq, where p and q are large primes (efficiently solving for s is equivalent to efficiently factoring n)

- Peggy uses a random number r , computes $x = r^2 t^{-1} \bmod n$ and sends x to Victor
- Victor asks either for the square root of x, or the square root of xt
- If Victor asked for the square root of x, Peggy doesn't know s and must guess $y = rs^{-1}$. If he asked for the square root of xt, Peggy lets y = r. She now returns y to Victor
- Victor checks that $y^2 \equiv x$ or $y^2 \equiv xt \pmod n$ depending on what he asked for

If Peggy can do this enough times (she must use different random numbers r each time), Victor will conclude that Peggy knows s, the square root of t

The Feige-Fiat-Shamir identification scheme

Peggy claims to know square roots s_i of t_i mod n = pq, where p and q are large primes

- Peggy uses a random number r , computes x = r^2 mod n and sends x to Victor
- Victor asks for the square root of $xt_1^{b_1}t_2^{b_2} \ldots t_k^{b_k}$ for $b_i \in \{0, 1\}$
- Peggy computes $y = rs_1^{b_1}s_2^{b_2} \ldots s_k^{b_k} \, mod \, n$ and sends y to Victor
- Victor checks $y^2 = r^2 s_1^{2b_1} s_2^{2b_2} \ldots s_k^{2b_k} \equiv xt_1^{b_1}t_2^{b_2} \ldots t_k^{b_k} \, (mod \, n)$

If Peggy can do this m times (she must use different random numbers r each time), Victor will conclude that Peggy knows s_i . The probability of Peggy correctly guessing what Victor's b_i will be is $1/2^{mk}$.

Use FFS to identification

Peggy is a customer at Victor's bank. She wants to identify herself electronically. To generate numbers to be used in the scheme, Peggy and Victor agree on the following:

- Peggy uses her name, Id, publicly known hash function, she also chooses (secret) p and q and gives Victor n = pq (as in RSA)
- She concatenates a counter to the end of the data, and hashes the resulting number, for a few values of the counter. Victor does the same
- Roughly half the numbers have square roots mod p, and half the numbers mod q. This means that roughly a quarter of the numbers have square roots mod n. Knowing p and q, Peggy can efficiently calculate these

They can now use Feige-Fiat-Shamir identication. An eavesdropper on a card transaction will learn nothing about the square roots that Peggy uses.

Zero-knowledge, formal setup

Peggy knows the solution to a hard mathematical problem, such as the pre-image to a one-way function output

- Peggy uses a random number to transform the problem into a different, but equally hard mathematical problem. She commits to the solution of the new instance (using bit commitment) and reveals it to Victor
- Victor asks Peggy to either
  a) prove that the two problems are isomorphic, or
  b) provide the solution of the second problem
- If Peggy can do this enough times, Victor will conclude that Peggy can solve the original problem

It is important that Peggy's and Victor's choices is random (to the other participant), otherwise cheating is possible

Digital signature are not Zero-knowledge

Either Alice chooses the message, and gives Bob message and signature. In this case, there is no (random) challenge from Bob.

- Bob is not convinced. Messages have no structure here, so how does he know Alice did not select the signature first and used the public key to generate the message?

Or Bob chooses the message and asks Alice to sign, but then, there is no (random) commitment by Alice

- Here the danger is that Bob could carefully select messages to extract information on the secret

In proper zero knowledge protocols, new randomness is added at each instance, so that no information on the secret is revealed to the verifier

Ďakujem za pozornosť.

jozef.jirasek@upjs.sk