

Short Notes for Zero Knowledge (WS 2008/09)

Last Update: February 16, 2009

Important note: These notes are not supposed to be self-contained. Instead, they are intended as a reminder about which topics were discussed in the lecture. If you find mistakes in these notes, please send them to unruh@mmci.uni-saarland.de.

Much of the content of this script is also covered in [Gol01, Chapter 4]. Be aware, however, that Goldreich might use slightly different notation or definitions in some cases.

Contents

1	Basic Definitions	1	
2	Graph-Isomorphism	4	
3	Amplification and sequential composition	5	
4	Zero-knowledge proofs for all NP-relations	7	
5	Composition of computational zero-knowledge	11	
6	Proofs of knowledge	11	
7	Efficient proof systems	12	
8	Combining efficient proofs	16	
9	Non-interactive zero-knowledge proofs	18	
10	Chosen-ciphertext secure encryption	20	Lecture on 2008-10-29

1 Basic Definitions

We call a function f *negligible*, if for every positive polynomial p , we have that $|f(n)| \leq \frac{1}{p(n)}$ for sufficiently large n . We call a function f *noticeable*, if there is a positive polynomial p such that for sufficiently large n , we have $f(n) \geq \frac{1}{p(n)}$. Note that no function can be both negligible and noticeable, but there are functions that are neither noticeable nor negligible (e.g., $f(2n) := 0$, $f(2n + 1) := 1$). We call a function *non-negligible* if it is not negligible. We call a function f *overwhelming* if $f \geq 1 - g$ for some negligible g .

In this lecture, we will often use the concept of an interactive machine. An *interactive machine* M is a machine that expects some input at the beginning of its execution, and from then on alternately sends and receives messages. It may finally terminate with some output. Such a machine may be probabilistic (i.e., use randomness). In some cases, we will assume that the randomness is given in the form of an infinite sequence of random bits, the *random tape*. Given two interactive machines P and V , we can define an execution of P and V on inputs x and y as follows: First P and V are started with input x and y , respectively. Then P is supposed to send a message. This message is given to V who in turn may respond with a message. This continues until V terminates with some output o . We then write $\langle P(x), V(y) \rangle$ for the random variable (or probability distribution¹) describing the output o of V .

We call an interactive machine M *polynomial-time* if there is a polynomial p such that upon input x , the machine M runs at most $p(|x|)$ steps (where $|x|$ is the length of x), no matter what and how many messages it receives. We call a machine M polynomial-time in its first input if for any input (x, y) , it runs at most $p(|x|)$ steps.

We will often use relations R on pairs of bitstrings. If $(x, w) \in R$ (or xRw), we say that x is a true statement and w is a witness for x . For example, if we want to prove that a number is composite (a product of nontrivial factors), we would use the relation R with $xR(p, q)$ iff $x = pq$ and $p, q > 1$. Intuitively, the witness w corresponds to a mathematical proof that x is a true statement. We write L_R for the language of all true statements, i.e., $L_R := \{x : \exists w. (x, w) \in R\}$.

Using this notation, we can define what it means for interactive machines to perform a proof:

Definition 1 (Proof system) An (interactive) proof system (or proof for short) for a relation R with completeness bound c and soundness bound s is a pair of polynomial-time² interactive machines P and V such that the following properties are fulfilled:

- **Completeness:** If $(x, w) \in R$ then $\Pr[\langle P(x, w), V(x) \rangle = 1] \geq c(|x|)$. (Intuitively, the honest prover will succeed in proving a true statement with probability at least $c(|x|)$, provided it knows a witness.)
- **Soundness:** For any (possibly computationally unbounded) interactive machine P^* (the cheating prover), and for any $x \notin L_R$, we have that $\Pr[\langle P^*, V(x) \rangle = 1] \leq s(|x|)$. (Intuitively, even a dishonest, unbounded prover will not succeed in convincing the honest verifier of a wrong statement.)

Obviously, a proof is better if c is close to 1 (it almost always succeeds), and s is close to 0 (one almost never proves anything wrong). Typically one either requires that c is overwhelming and s is negligible, or that $c \geq \frac{2}{3}$ and $s \leq \frac{1}{3}$. Obviously, in the second case the proof is not terribly useful, yet as we will see later, a proof with $c \geq \frac{2}{3}$ and $s \leq \frac{1}{3}$

¹In this text, we will not distinguish between random variables and distributions, and—somewhat informally—use the terms interchangeably.

²Often, one only requires V to be polynomial-time. In this lecture, however, both V and P will always be polynomial-time unless mentioned otherwise.

can easily be converted into one with overwhelming c and negligible s (amplification). See Theorem 2.

If $c = 1$, we say the proof has *perfect completeness*. Many proofs have this property.

For the definition of zero-knowledge, we first need the following definitions that allow to quantify how close two probability distributions are:

Definition 2 (Statistical distance) *Given two random variables (or distributions) X and Y , the statistical distance $\text{SD}(X; Y)$ between X and Y is defined as*

$$\text{SD}(X; Y) := \max_T |\Pr[X \in T] - \Pr[Y \in T]|$$

where T ranges over all sets of values that X and Y might take (i.e., if the random variables X and Y take natural numbers as values, then T ranges over all subsets of \mathbb{N}).³

The intuition behind this definition is that, if two random variables can be distinguished, this can be done by some statistical test T getting a sample from one of the distributions and then guessing whether this sample was drawn from X or from Y . Such a test is modelled as a set (then T would contain, e.g., all samples where we would give the answer “ X ”). Then $|\Pr[X \in T] - \Pr[Y \in T]|$ tells us how well T distinguishes between X and Y , and $\text{SD}(X; Y)$ tells us, how well the best possible test would distinguish X and Y .

Lemma 1 (Properties of the statistical distance) *Let X , Y , and Z be random variables. Let f be a function. Then the following properties hold for the statistical distance:*

- $\text{SD}(X; Y) = \text{SD}(Y; X)$ (symmetry).
- $\text{SD}(X; Y) \geq 0$ (nonnegativity)
- $\text{SD}(X; Y) = 0$ iff X and Y have the same distribution (nondegeneracy)
- $\text{SD}(X; Z) \leq \text{SD}(X; Y) + \text{SD}(Y; Z)$ (triangle inequality)
- SD is a metric (this is equivalent to the four properties above).
- $\text{SD}(X; Y) \leq 1$.
- $\text{SD}(X; Y) = \frac{1}{2} \sum_a |\Pr[X = a] - \Pr[Y = a]|$ (as long as the range of X and Y is countable).
- $\text{SD}(f(X); f(Y)) \leq \text{SD}(X; Y)$ (no computation can increase the statistical distance)
- If Z is stochastically independent of X and of Y , then $\text{SD}((X, Z); (Y, Z)) = \text{SD}(X; Y)$ (independent additional information does not change the statistical distance).

³To be exact, T ranges only over measurable sets. But in all cases considered in this lecture, there are no nonmeasurable subsets of the set of the range of X and Y , so you may always assume that T ranges over all sets. (If you do not know what this footnote means, just ignore it.)

We can now formulate what it means that a proof does not leak any information by modelling the fact that anything that could be output by a cheating verifier can also be output by some simulator that does not know any secret data.

Definition 3 (Statistical zero-knowledge without auxiliary input) A pair (P, V) of interactive machines is called statistically zero-knowledge without auxiliary input for a relation R if for any polynomial-time machine V^* (the cheating verifier), there exists a polynomial-time algorithm S (called the simulator), and a negligible function μ such that for all $(x, w) \in R$ we have that

$$\text{SD}(\langle P(x, w), V^*(x) \rangle; S(x)) \leq \mu(|x|).$$

Note that the simulation produced by the simulator does not need to be perfect (as this is rarely possible), but instead only has to be very close to the distribution in the real proof performed by P and V^* .

2 Graph-Isomorphism

In the setting of this section, a graph is modelled as a set $V = \{1, \dots, n\}$ of vertices and a set $E \subseteq V \times V$ of edges where $(x, y) \in E$ iff $(y, x) \in E$ (i.e., the graph is undirected). We write $|G|$ for the number of vertices of G .

A graph isomorphism is a permutation ϕ on $\{1, \dots, n\}$ (for some n). If we apply ϕ to a graph $G = (V, E)$ with $V = \{1, \dots, n\}$, the resulting graph $\phi(G) = (\phi(V), \phi(E))$ is defined by $\phi(V) = V$ and $\phi(E) = \{(\phi(x), \phi(y)) : (x, y) \in E\}$. In other words, a graph isomorphism reorders the vertices of a graph and moves the edges correspondingly.

Two graphs G and H are isomorphic if there exists a graph isomorphism ϕ such that $H = \phi(G)$.

It is considered to be hard to find out whether two graphs are isomorphic.

Definition 4 (ZK-Proof for Graph Isomorphism) The ZK-proof system (P, V) for graph isomorphism is defined as follows:

- *Statement:* $x = (G_1, G_2)$ where G_1 and G_2 are graphs.
- *Witness:* $w = \phi$ where ϕ is a graph isomorphism and $G_2 = \phi(G_1)$.
- *Prover:* On input $x = (G_1, G_2)$ and $w = \phi$, the prover P chooses a uniformly random graph isomorphism τ on G_1 (i.e., it chooses a random permutation on $\{1, \dots, |G_1|\}$). Then it computes $H := \tau(G_2)$. Then it sends H to V .
- *Verifier:* On input $x = (G_1, G_2)$ and when receiving a graph H from the prover, the verifier chooses $i \in \{1, 2\}$ uniformly at random and sends i to the prover.
- *Prover:* When receiving $i \in \{1, 2\}$ from the verifier V , if $i = 1$, the prover P sends $\tilde{\phi} := \tau \circ \phi$ to V , and if $i = 2$, the prover P sends $\tilde{\phi} := \tau$ to V .

- *Verifier:* When receiving $\tilde{\phi}$ from the prover, the verifier V checks whether $\tilde{\phi}(G_i) = H$. If so, it outputs 1. Otherwise it outputs 0.

Theorem 1 *The ZK-proof system for graph isomorphism has the following properties:*

- *It has perfect completeness.*
- *It has soundness bound $\frac{1}{2}$.*
- *It is statistically zero-knowledge.*

For the zero-knowledge property, the simulator S for some verifier V^* would be the following:

- Input: $x = (G_1, G_2)$, $z \in \{0, 1\}^*$.
- Choose i randomly.
- Choose a graph isomorphism $\tilde{\phi}$.
- Compute $H = \tilde{\phi}(G_i)$.
- Invoke the (simulated) verifier V^* on input (x, z) and send the message H to V^* (as coming from the prover).
- Expect $i^* \in \{1, 2\}$ from V^* .
- If $i = i^*$, send $\tilde{\phi}$ to V^* . Output whatever V^* outputs.
- If $i \neq i^*$, restart the simulation from the beginning. (But at most $|x|$ times, otherwise S would not be polynomial-time in its first input.)

Lecture on
2008-11-12

3 Amplification and sequential composition

To be useful, a proof system should have completeness bound close to 1 and soundness bound close to 0. Our definition of proof systems (Definition 1), however, allows for much worse completeness and soundness bounds. Hence, one needs a method for improving the soundness and completeness bound of a proof system. The following theorem shows that this is possible as long as there is a small gap between the soundness and the completeness bound.

Definition 5 *Let (P, V) be a proof system. Let $f : \mathbb{N} \rightarrow \mathbb{N}$ be a positive function. Let $g : \mathbb{N} \rightarrow [0, 1]$ be a function. We define P^f to be the interactive machine which on input x, w runs $P(x, w)$ $f(|x|)$ times sequentially.⁴ We define V_g^f to be the machine that on input x runs $V(x)$ $f(|x|)$ times. Let t be the number of instances of V (run by V_g^f) that return 1. Then V_g^f returns 1 iff $t/f(|x|) \geq g(|x|)$.*

⁴That is, P^f runs $P(x)$. When $P(x)$ finishes, P^f runs $P(x)$ again. Etc. $f(|x|)$ times.

Theorem 2 (Amplification of proofs) *Let (P, V) be a proof system with completeness bound c and soundness bound s . Assume that $c - s$ is noticeable. Then there is a positive polynomial f such that the following holds: $(P^f, V_{\frac{c+s}{2}}^f)$ has soundness bound $2^{-|x|}$ and completeness bound $1 - 2^{-|x|}$.*

If (P, V) has perfect completeness, then $(P^f, V_{\frac{c+s}{2}}^f)$ has perfect completeness, too.

(A special case of this construction was proven on the first exercise sheet.)

If we are interested in zero-knowledge proofs, however, it is not sufficient that the resulting proof has good soundness and completeness. We also want that, if (P, V) is zero-knowledge, then $(P^f, V_{\frac{c+s}{2}}^f)$ is also zero-knowledge. For the notion of zero-knowledge put forth in Definition 3, it is not clear whether this holds. Instead, we need to use the following modified definition.

Definition 6 (Statistical zero-knowledge with auxiliary input) *A pair (P, V) of interactive machines is called statistically zero-knowledge with auxiliary input for a relation R if for any machine V^* (the cheating verifier) polynomial-time in its first input, there exists an algorithm S (called the simulator) polynomial-time in its first input, and a negligible function μ such that for all $(x, w) \in R$ and all bitstrings $z \in \{0, 1\}^*$ (the auxiliary input) we have that*

$$\text{SD}(\langle P(x, w), V^*(x, z) \rangle; S(x, z)) \leq \mu(|x|).$$

In the literature, statistical zero-knowledge typically denotes statistical zero-knowledge with auxiliary input.

For this definition, we can prove the so-called *sequential composition theorem*:

Theorem 3 (Sequential composition theorem) *Assume that (P, V) is statistically zero-knowledge with auxiliary input (Definition 6). Then for any efficiently computable, polynomially-bounded function f , and for any function g , we have that (P^f, V_g^f) is statistically zero-knowledge with auxiliary input.*

We have shown the sequential composition theorem with the aim of amplification in mind. However, the fact that zero-knowledge proofs compose sequentially has much more far reaching consequences. For example, it is important that zero-knowledge proofs keep their zero-knowledge property even when used as part of a larger protocol. If in the larger protocol, the zero-knowledge proof is used sequentially (i.e., the larger protocol pauses while the ZK proof is executed), then usually the ZK proof still behaves well in that context (this does not follow from Theorem 3, but in most cases it is shown using similar techniques).⁵ In most cases, the auxiliary input is essential for such compositionality results. Note that the sequential input usually only helps for sequential composition. If several proofs are allowed to run in parallel or concurrently, composition is much more difficult to achieve.

⁵Of course, what it means to behave well depends on the security notion of the overall protocol and needs to be checked on a case-by-case basis.

A general rule of thumb (not specific to ZK) that should be remembered whenever some kind of composition is an issue:

If something might already have happened before the execution of a protocol π , then usually the protocol π needs to be secure with respect to some security notion that uses auxiliary input.

(There are exceptions, however. For example the so-called Universal Composability framework (UC) works nicely without auxiliary input.)

4 Zero-knowledge proofs for all NP-relations

Definition 7 (NP-relation) A relation R over bitstrings is an NP-relation if

- there is a deterministic polynomial-time algorithm that on input (x, w) decides whether $(x, w) \in R$ and
- there is a polynomial p such that for all $(x, w) \in R$, we have $|w| \leq p(|x|)$. (poly-balanced)

Lemma 2 A language L is in NP if and only if there is an NP-relation R such that $L = L_R$.

Definition 8 (NP-complete relations) A relation R is NP-complete^{6,7} if

- R is an NP-relation and
- for any NP-relation R' there are deterministic polynomial-time functions f, g, h such that
 - $\forall x. x \in L_{R'} \iff f(x) \in L_R$.
 - $\forall (x, w) \in R'. (f(x), g(x, w)) \in R$. (If you have a witness w.r.t. R' , you have one w.r.t. R .)
 - $\forall x. (f(x), w) \in R \implies (x, h(w)) \in R'$. (If you have solved the problem in R , you can carry the witness back to R' .)

Theorem 4 Let R be an NP-complete relation. Let R' be an NP-relation. Let f, g, h be as in Definition 8.

Given a prover P , let P' be the prover that on input (x, w) runs $P(f(x), g(x, w))$. Given a verifier V , let V' be the verifier that on input x runs $V(f(x))$.

If (P, V) is a statistical zero-knowledge proof for R with/without auxiliary input with overwhelming completeness bound and negligible soundness bound, then (P', V') is a statistical zero-knowledge proof for R' with/without auxiliary input with overwhelming completeness bound and negligible soundness bound.⁸

⁶Usually, one does not speak about NP-complete relations, but about NP-complete search problems. For uniformity of language, I use the word relation here.

⁷As far as I know, there is no single universally accepted definition of NP-completeness. Hence other authors may have different notions of NP-completeness.

⁸However, the exact soundness bounds may change because they are measured in terms of the length of the statement, which is $|x|$ in one case and $|f(x)|$ in the other case.

Definition 9 (Graph-3-colouring) Let a graph $G = (V, E)$ with vertices V and edges E be given. Let a mapping $\gamma : V \rightarrow \{R, G, B\}$ be given. We call γ a 3-colouring of G if for all $(v_1, v_2) \in E$ we have $\gamma(v_1) \neq \gamma(v_2)$.

We define R_{G3C} by $(G, \gamma) \in R_{G3C}$ iff γ is a 3-colouring of G .

Lemma 3 R_{G3C} is an NP-complete relation.

Proofs will be found in most textbooks on complexity theory.

Hence, if we can find a zero-knowledge proof for R_{G3C} , by Lemma 3 and Theorem 4 we get zero-knowledge proofs for an NP-relation. Unfortunately, it is widely believed that there is no statistical zero-knowledge proof for any NP-complete relation.⁹ However, below we will see a relaxed definition of zero-knowledge (computational zero-knowledge) that allows us to prove that the proof system for R_{G3C} described below is ZK.

But first, we need to introduce the concept of a commitment.

Definition 10 A perfectly binding computationally hiding non-interactive commitment (short: commitment) scheme consists of a probabilistic polynomial-time algorithm Com and a deterministic polynomial-time algorithm $Verify$ and a sequence M_n of sets (the message space that depends on the security parameter) such that:

- *Correctness:* For all $n \in \mathbb{N}$ and for all $m \in M_n$, we have that $\Pr[Verify(1^n, m, c, u) = 1 : (c, u) \leftarrow Com(1^n, m)] = 1$ and for all $m \notin M_n$ and all $c, u \in \{0, 1\}^*$, we have $Verify(1^n, m, c, u) = 0$.
- *Perfectly binding:* For all $n \in \mathbb{N}$ and all $c \in \{0, 1\}^*$ there exists at most one $m \in \{0, 1\}^*$ such that there exists some $u \in \{0, 1\}^*$ such that $Verify(1^n, m, c, u) = 1$.
- *Computationally hiding:* For all nonuniform probabilistic polynomial-time algorithms B^* there exists a negligible function μ such that for all $n \in \mathbb{N}$ and all $m_1, m_2 \in M_n$ we have that

$$\left| \Pr[B^*(1^n, c) = 1 : (c, u) \leftarrow Com(1^n, m_1)] - \Pr[B^*(1^n, c) = 1 : (c, u) \leftarrow Com(1^n, m_2)] \right| \leq \mu(n).$$

Theorem 5 If one-way permutations exist then commitment schemes (in the sense of Definition 10) exist.¹⁰

Definition 11 (G3C proof) We define the following proof system (P_{G3C}, V_{G3C}) for R_{G3C} .

- Fix a commitment scheme with message space $M_n = \{R, G, B\}$.
- The statement is a graph $G = (V, E)$.
- The witness is a 3-colouring γ of G .
- The prover chooses a random permutation ρ on $\{R, G, B\}$.

⁹For those who know what this means: If there were such proofs, the polynomial hierarchy would collapse.

¹⁰As long as $m \in M_n$ is efficiently decidable and the length of m is polynomially bounded in n .

- For each $v \in V$, the prover computes $(c_v, u_v) \leftarrow \text{Com}(\rho(\gamma(v)))$.
- The prover sends all c_v ($v \in V$) to the verifier.
- The verifier randomly chooses $(v_1, v_2) \in E$ and sends (v_1, v_2) to P .
- The prover checks whether $(v_1, v_2) \in E$. Then he sets $m_1 := \rho(\gamma(v_1))$ and $m_2 := \rho(\gamma(v_2))$ and sends $(m_1, m_2, u_{v_1}, u_{v_2})$ to the verifier.
- The verifier checks whether $m_1 \neq m_2$ and $\text{Verify}(m_1, u_{v_1}) = 1$ and $\text{Verify}(m_2, u_{v_2}) = 1$.

Theorem 6 (P_{G3C}, V_{G3C}) has perfect completeness and soundness bound $1 - \frac{1}{|E|}$ for R_{G3C} (where E is the set of edges of the statement).¹¹

Note that the soundness bound can be improved using amplification (Theorem 2).

Although (P_{G3C}, V_{G3C}) is a proof system, it is not statistically zero-knowledge. The intuitive reason is that the commitments are only computationally hiding, hence the witness is information-theoretically leaked (see also exercise sheet 4, problem 2b). We therefore need to weaken the definition of zero-knowledge to allow that the simulation is not statistically close to the real proof, only seems so to computationally bounded machines.¹²

**Lecture on
2008-11-25**

Definition 12 (Computational indistinguishability) Let two families of distributions $\{X_{x,a}\}_{x,a}$ and $\{Y_{x,a}\}_{x,a}$ be given. Let a set A be given. Then X and Y are computationally indistinguishable for all $(x, a) \in A$ if for every algorithm D polynomial-time in its first input there is a negligible function μ such that for all $(x, a) \in A$ we have that

$$\left| \Pr[D(x, a, X_{x,a}) = 1] - \Pr[D(x, a, Y_{x,a}) = 1] \right| \leq \mu(|x|).$$

Note that in the literature, the definition of computational indistinguishability varies in the details (and is not always equivalent to our definition in all cases), but the basic theme is always like in Definition 12.

For comparison, the following notion of indistinguishability is implicit in Definition 6.

Definition 13 (Statistical indistinguishability) Let two families of distributions $\{X_{x,a}\}_{x,a}$ and $\{Y_{x,a}\}_{x,a}$ be given. Let a set A be given. Then X and Y are statistically indistinguishable for all $(x, a) \in A$ if there is a negligible function μ such that for all $(x, a) \in A$ we have that

$$\text{SD}(X_{x,a}; Y_{x,a}) \leq \mu(|x|).$$

¹¹If we want to strictly follow our notation that the soundness is measured in the size of the witness $x = G$, then we can instead use the somewhat worse soundness bound $1 - \frac{1}{|x|}$ because the size of the representation of a graph is larger than the number of edges it has.

¹²Alternatively, we could also implement the G3C proof using statistically hiding commitments. But these would necessarily be only computationally binding, hence the G3C proof system would not be a proof (as an unbounded prover could cheat), but only a so-called argument or computationally sound proof (that is secure against polynomial-time provers). So we would still have to change one definition.

Note that if we define statistical indistinguishability like computational indistinguishability, except that D is allowed to be unbounded,¹³ then the resulting definition is equivalent to Definition 13.

Given Definition 12, we can formulate the notion of computational zero-knowledge:

Definition 14 (Computational zero-knowledge) *A pair (P, V) of interactive machines is called computationally zero-knowledge with auxiliary input for a relation R if for any machine V^* polynomial-time in its first input, there exists an algorithm S polynomial-time in its first input such that the following two families of distributions are computationally indistinguishable for all $x, w, z \in \{0, 1\}^*$ with $(x, w) \in R$:*

$$\left\{ \langle P(x, w), V^*(x, z) \rangle \right\}_{x, (w, z)} \quad \text{and} \quad \left\{ S(x, z) \right\}_{x, (w, z)}$$

Usually, we do not mention the “with auxiliary input”. Computational zero-knowledge *without* auxiliary input is defined analogously to Definition 3 by removing all occurrences of z from Definition 14.

Now we can formulate the fact that the G3C-proof system is zero-knowledge:

Theorem 7 $(P_{\text{G3C}}, V_{\text{G3C}})$ *is computationally zero-knowledge with auxiliary input for the relation R_{G3C} .*

The simulator S for a malicious verifier V^* is constructed as follows: Upon input (x, z) with $x = G = (V, E)$, S runs the simulator $S_1(x, z)$ until $S_1(x, z)$ succeeds (at most $|E|^2$ times) and outputs its output. The simulator S_1 is defined as follows (for simplicity, we assume that V^* never sends $(v_1, v_2) \notin E$):

- Simulate $V^*(x, z)$.
- Choose $(v_1, v_2) \in E$ at random.
- Choose $m_{v_1}, m_{v_2} \in \{R, G, B\}$ at random under the condition that $m_{v_1} \neq m_{v_2}$.
- Set $m_v = R$ for all $v \in V$ with $v \neq v_1, v_2$.
- Compute $(c_v, u_v) \leftarrow \text{Com}(m_v)$ for all $v \in V$.
- Send all c_v to V^* . Let v_1^*, v_2^* be the response.
- Abort if $(v_1, v_2) \neq (v_1^*, v_2^*)$.
- Send $(m_{v_1}, m_{v_2}, u_{v_1}, u_{v_2})$ to V^* . Let *out* denote the output of V^* .
- Return *out*.

Using the fact that R_{G3C} is NP-complete (Lemma 3), we get the following corollary.

Corollary 1 *If R is an NP-relation, then there is a computationally zero-knowledge proof system (P, V) for R with polynomial-time P and V , with perfect completeness, and with soundness bound $1 - 1/p(|x|)$ for some positive unbounded polynomial p .*

¹³Here an unbounded algorithm is assumed to be able to perform any computation, even uncomputable ones.

5 Composition of computational zero-knowledge

With the notation from Section 3, we have the following analogue of Theorem 3 for computational zero-knowledge.

Theorem 8 (Sequential composition theorem) *Assume that (P, V) is computational zero-knowledge with auxiliary input (Definition 6). Then for any efficiently computable, polynomially-bounded function f , and for any function g , we have that (P^f, V_g^f) is computational zero-knowledge with auxiliary input.*

Combining Corollary 1, Theorem 8, and Theorem 2, we get the following corollary.

Corollary 2 *If R is an NP-relation, then there is a computationally zero-knowledge proof system (P, V) for R with polynomial-time P and V , with perfect completeness, and with soundness bound $2^{-|x|}$.*

6 Proofs of knowledge

Proofs in the sense of Definition 1 guarantee that no statement x is accepted by the verifier such that there is no witness w with $(x, w) \in R$. Such a proof can hence be interpreted as proving “there is a witness w for x ”. In some cases, however, we need the prover to show something stronger, namely: “I *know* a witness w for x ”. This is captured by the notion of a *proof of knowledge* which is defined below.

In the following, when we say a machine K has *oracle access with rewinding* to an interactive machine P , we mean the following: The machine K may send inputs of the following form to P : (msg, m) and $(rewind, i)$. Upon input (msg, m) , the code of P is executed on the current state of P to compute the reaction of P to the message m . The message that is output by P is returned to K . The new state of P is appended to a list H of states. Upon input $(rewind, i)$, the state of P is set to the i -th element of H .

Summarising, K can interact with P in an arbitrary fashion and at any point reset P 's state to any state in the history H of the execution.

Definition 15 *We call (P, V) a proof of knowledge with completeness bound c , soundness bound s , and knowledge error κ if the following holds:*

- (P, V) is a proof with completeness bound c , soundness bound s .
- Validity: *There exists a constant $d > 0$ and a polynomial-time¹⁴ oracle machine K with rewinding such that for any interactive machine P^* and any $x \in L_R$, we have the following:*

$$\Pr[\langle P^*, V(x) \rangle = 1] \geq \kappa(|x|) \implies \Pr[(x, w) \in R : w \leftarrow K^{P^*}(x)] \geq (\Pr[\langle P^*, V(x) \rangle = 1] - \kappa(|x|))^d.$$

¹⁴In the case of an oracle machine K , for the definition of polynomial-time we only count computation steps made by the oracle machine K itself, not the steps performed by the oracle P^* .

An example for a proof of knowledge is the graph isomorphism proof (Definition 4), see exercise sheet 6, problem 2.

Note that our definition of proofs of knowledge differs from that of [Gol01]. There are actually many possible variants on how to define proofs of knowledge. Yet, all have the basic idea of an extractor that tries to extract a witness from the prover. Moreover, the idea of an extractor to model the notion of some party having to know something is not restricted to zero-knowledge. For example, in definitions of secure multi-party computation, such an extractability property is often explicitly or implicitly required.

Theorem 9 (Amplification of knowledge error) *Let p be a positive polynomial. Assume that (P, V) has knowledge error $\kappa \leq 1 - 1/p$. Then there exists a positive polynomial f such that (P^f, V_1^f) has knowledge error 0. (Notation as in Definition 5.)*

The proof system for graph-3-colouring has knowledge error $\kappa \leq 1 - 1/|E|$ where E is the number of edges of the graph. Hence we get the following corollary:

Corollary 3 *If R is an NP-relation, then there is a computationally zero-knowledge proof of knowledge (P, V) for R with polynomial-time P and V , with perfect completeness, and with soundness bound $2^{-|x|}$, and with knowledge error 0.*

But again, the construction we derived is highly inefficient.

7 Efficient proof systems

We will now turn our attention to efficient proof systems.

The following proof system proves the knowledge of a discrete logarithm:

Definition 16 (Schnorr's proof system for discrete logarithms) *Fix a cyclic group G of prime order and a generator g of G (for simplicity, we assume that G and g depend on some implicitly given security parameter k). Let $R := \{(x, w) : x = g^w\}$. Let $N := \{0, \dots, \text{ord } G - 1\}$.*

Then (P, V) is defined as follows:

- *Input of prover P : $(x, w) \in R$.*
- *Input of verifier V : x .*
- *P chooses $b \xleftarrow{R} N$ and sends $a := g^b$ to V .*
- *V chooses $r \xleftarrow{R} N$ and sends r to P .*
- *P computes $s := b + rw \pmod{\text{ord } G}$ and sends s to V .*
- *V checks whether $x, a \in G$ and $g^s = ax^r$.*

This protocol has an important outer form that is important for many constructions:

Definition 17 (Σ -protocol) (P, V) is called a Σ -protocol if the interaction consists of three messages a, r, s (send by P, V, P , respectively), and r is uniformly chosen from some set N (that may only depend on x). (In particular, r is chosen independently of a .) Furthermore, the verifier decides whether to accept or not after the last message by a deterministic polynomial-time computation on x, a, r, s where x is the common input (the statement). Also sampling uniformly from N should be possible in polynomial-time, and verifying whether some $r \in N$ should be possible in deterministic polynomial-time.

Obviously, Schnorr's protocol is a Σ -protocol.

Theorem 10 Schnorr's protocol has perfect completeness, soundness $1/|N|$, and—if $|N|$ is superpolynomial—knowledge error 0.

Note however, that Schnorr's protocol is not known to be zero-knowledge.

Schnorr's protocol fulfils an even stronger notion of soundness/proof of knowledge:

Definition 18 (Special soundness) We say a Σ -protocol (P, V) has special soundness for a relation R if there is a deterministic polynomial-time algorithm E such that the following holds: For any two interactions (a, r, s) and (a, r', s') that $r \neq r'$ and $r, r' \in N$ and such that the verifier would accept both interactions, $w := E(x, a, r, s, r', s')$ satisfies $(x, w) \in R$. (Here x, a, r, s, N are as in Definition 17.)

Lemma 4 Schnorr's protocol has special soundness.

Lemma 5 If a Σ -protocol has special soundness, then it has soundness $1/|N|$, and—if $|N|$ is superpolynomial—knowledge error 0.

Although Schnorr's protocol is not zero-knowledge, it satisfies a weaker notion:

Definition 19 (Honest-verifier computational zero-knowledge) A pair (P, V) of interactive machines is called computationally honest-verifier zero-knowledge (HVCZK) for a relation R if there exists an algorithm S polynomial-time in its first input such that the following two families of distributions are computationally indistinguishable for all $x, w \in \{0, 1\}^*$ with $(x, w) \in R$:

$$\left\{ \text{view}\langle P(x, w), V(x) \rangle \right\}_{x, w} \quad \text{and} \quad \left\{ S(x) \right\}_{x, w}$$

The difference to Definition 14 is that the simulator only needs to simulate the view of the honest verifier. A dishonest verifier may learn as much as it pleases. Note that there is no sense in adding an auxiliary input in this definition, as the honest verifier would not use it anyway.

Although not necessarily useful on its own, HVCZK is a useful notion because there are efficient transformations for producing ZK proofs from HVCZK ones.

Schnorr's scheme fulfils an even stronger notion of HVCZK:

Definition 20 (Special HVCZK) A Σ -protocol (P, V) is called computationally special honest-verifier zero-knowledge (special HVCZK) for a relation R if there exists an algorithm S polynomial-time in its first input such that the following two families of distributions are computationally indistinguishable for all $x, w, r \in \{0, 1\}^*$ with $(x, w) \in R$ and $r \in \mathcal{N}$:

$$\left\{ \text{view}\langle P(x, w), V_r(x) \rangle \right\}_{x, w} \quad \text{and} \quad \left\{ S(x, r) \right\}_{x, w}$$

where V_r is a verifier that always sends r . We require that S outputs an interaction (a, r, s) with the same r as it got as input.

The difference to HVCZK is that the simulator is able to produce a simulation even if it is instructed to use a particular value of r .

We state the following straightforward fact:

Lemma 6 If a Σ -protocol is special HVCZK, then it is HVCZK.

Theorem 11 Schnorr's scheme is special HVCZK.¹⁵

Definition 21 (Equivocal trapdoor commitments) A computationally binding equivocal¹⁶ non-interactive trapdoor commitment (short: trapdoor commitment) scheme consists of a probabilistic polynomial-time algorithm KeyGen , a probabilistic polynomial-time algorithm Com , a probabilistic polynomial-time algorithm Equiv , and a deterministic polynomial-time algorithm Verify and a sequence M_n of sets (the message space that depends on the security parameter) with $0 \in M_n$ such that:

- **Correctness:** For all $n \in \mathbb{N}$ and for all $m \in M_n$, we have that $\Pr[\text{Verify}(1^n, \text{crs}, m, c, u) = 1 : (\text{crs}, \text{td}) \leftarrow \text{KeyGen}(1^n), (c, u) \leftarrow \text{Com}(1^n, \text{crs}, m)] = 1$ and for all $m \notin M_n$ and all $c, u \in \{0, 1\}^*$, we have $\text{Verify}(1^n, \text{crs}, m, c, u) = 0$.
- **Computationally binding:** For every non-uniform polynomial time algorithm A ,

$$\Pr[\text{Verify}(1^n, \text{crs}, m, c, u) = 1 \wedge \text{Verify}(1^n, \text{crs}, m', c, u') = 1 \wedge m \neq m' : (\text{crs}, \text{td}) \leftarrow \text{KeyGen}(1^n), (c, m, u, m', u') \leftarrow A(1^n, \text{crs})]$$

is negligible.

- **Equivocality:**¹⁷ The following two families of distributions are computationally indistinguishable for all $n \in \mathbb{N}, m \in M_n, z \in \{0, 1\}^*$:

$$\left\{ \begin{array}{l} (c, u) : \\ (\text{crs}, \text{td}) \leftarrow \text{KeyGen}(1^n) \\ (c, \tilde{u}) \leftarrow \text{Com}(1^n, \text{crs}, 0), \\ u \leftarrow \text{Equiv}(1^n, \text{td}, c, \tilde{u}, m) \end{array} \right\}_{(1^n, m), z} \quad \text{and} \quad \left\{ \begin{array}{l} (c, u) : \\ (\text{crs}, \text{td}) \leftarrow \text{KeyGen}(1^n) \\ (c, u) \leftarrow \text{Com}(1^n, \text{crs}, m) \end{array} \right\}_{(1^n, m), z}$$

¹⁵Schnorr's scheme even satisfies stronger notions like special honest-verifier statistical/perfect zero-knowledge. However, as we have not defined these here, we state the weaker results.

¹⁶Also known as "equivocable"

¹⁷Also known as "equivocability"

(Alternatively, one might also allow arbitrary messages $m' \in M_n$ instead of 0 as argument for Com on the left hand side. Then $Equiv$ also needs to get m' as additional argument. This leads to a stronger definition. For our purposes, this is not necessary.)

Definition 22 (Computational zero-knowledge in the CRS model) A triple $(P, V, KeyGen)$ of interactive machines is called computationally zero-knowledge with auxiliary input in the CRS model for a relation R if for any machine V^* polynomial-time in its first input, there exists an algorithm S polynomial-time in its first input such that the following two families of distributions are computationally indistinguishable for all $x, w, z \in \{0, 1\}^*$ with $(x, w) \in R$:

$$\left\{ \langle P(x, crs, w), V^*(x, crs, z) \rangle : crs \leftarrow KeyGen(1^{|x|}) \right\}_{x, (w, z)} \quad \text{and} \quad \left\{ S(x, z) \right\}_{x, (w, z)}$$

(One could also define this with $KeyGen$ additionally outputting a trapdoor. However, since the trapdoor is never used, we do not require $KeyGen$ to output a trapdoor (it may compute and then erase it, though).)

Definition 23 (Argument system) We call $(P, V, KeyGen)$ an argument¹⁸ in the CRS model for a relation R if $P, V, KeyGen$ are polynomial time and the following properties are fulfilled:

- **Completeness:** There is a negligible function μ such that for all $(x, w) \in R$ we have $\Pr[\langle P(x, crs, w), V(x, crs) \rangle = 1 : crs \leftarrow KeyGen(1^{|x|})] \geq 1 - \mu(|x|)$.
- **Computational soundness:** For any interactive machine P^* polynomial in its first input there is a negligible function μ such that for $x \notin L_R$ and any $z \in \{0, 1\}^*$, we have that $\Pr[\langle P^*(x, crs, z), V(x, crs) \rangle = 1 : crs \leftarrow KeyGen(1^{|x|})] \leq \mu(|x|)$.¹⁹

For simplicity, we have not parametrised this definition over the soundness bound and the completeness bound. There is, however, no principal reason to do so. To analyse results analogous to Theorem 2, one would of course need to add these parameters.

(One can derive the definition of an argument that is not in the CRS model by just removing all occurrences of $KeyGen$ and of the CRS from the definition.)

Definition 24 (Argument of Knowledge) We call $(P, V, KeyGen)$ an argument of knowledge in the CRS model if the following holds:

- $(P, V, KeyGen)$ is an argument (Definition 23).
- **Computational validity:** There exists a constant $d > 0$ and a polynomial-time oracle machine K with rewinding such that for any interactive machine P^* polynomial in its first input there is a negligible function μ such that for any $x \in L_R$,

¹⁸Also called “computationally sound proof”

¹⁹Note that here we add explicit arguments x and z to the prover because in contrast to an unlimited prover, a polynomial P^* could not “just know” these values.

$z \in \{0,1\}^*$, we have the following:

$$\begin{aligned} \Pr[(x, w) \in R : w \leftarrow K^{P^*(x, \cdot, z)}(x)] \\ \geq (\Pr[\langle P^*(x, crs, z), V(x) \rangle = 1 : crs \leftarrow KeyGen(1^{|x|})])^d - \mu(|x|). \end{aligned}$$

Here K is allowed to rewind P^* and is allowed to choose the CRS-argument passed to P^* (but not the arguments x and w).²⁰

Construction 1 (CZK from special HVCZK [Dam00]) Let (P, V) be a Σ -protocol. Let $(KeyGen, Com, Equiv, Verify)$ be an equivocal trapdoor commitment (Definition 21).

We define the following Σ -protocol $(P', V', KeyGen)$ in the CRS model (notation as in Definition 17):

- Input of P' : Statement x , CRS crs , Witness w . $n := |x|$.
- Input of V' : Statement x , CRS crs .
- P' simulates $P(x, w)$. V' simulates $V(x)$.
- P' asks P for a and computes $(c, u) \leftarrow Com(1^n, crs, a)$. Then P' sends c .
- V' chooses $r \xleftarrow{R} N$ and sends r .
- P' passes r to P and asks for s .
- P' sends (s, a, u) to V' .
- V' accepts if V would accept (a, r, s) and $Verify(1^n, crs, a, c, u) = 1$.

Theorem 12 Let (P, V) be a Σ -protocol. Let $(KeyGen, Com, Equiv, Verify)$ be an equivocal trapdoor commitment (Definition 21). Assume that for any statement x , the first message a sent by the prover is in $M_{|x|}$ where M_n is the message space of Com . Assume that (P, V) has overwhelming completeness, special soundness, is special HVCZK, and that $|N|$ is superpolynomial (where N is the set from which the verifier chooses its message r).

Let (P', V') be as in Construction 1. Then (P', V') is a computational zero-knowledge argument of knowledge in the CRS model (Definitions 22, 23, 24).

As there are efficient trapdoor commitment schemes, as well as efficient Σ -protocols for many relations for group-based problems, Construction 1 allows to construct very efficient ZK proofs in the CRS model.

**Lecture on
2009-01-21**

8 Combining efficient proofs

In the following, given two relations R_1 and R_2 and two sets M_1 and M_2 of bitstrings, let $R_1 \vee_{M_1, M_2} R_2 := \{((x_1, x_2), (i, w)) : i \in \{1, 2\}, (x_i, w) \in R_i \wedge x_1 \in M_1 \wedge x_2 \in M_2\}$ and $R_1 \wedge R_2 := \{((x_1, x_2), (w_1, w_2)) : (x_1, w_1) \in R_1, (x_2, w_2) \in R_2\}$.

²⁰In Construction 1 below, we actually do not need that K can choose the CRS. We fulfil the stronger definition where K is able to extract if crs having $\Pr[(x, w) \in R : w \leftarrow K^{P^*(x, \cdot, z)}(x) : crs \leftarrow KeyGen(1^{|x|})]$ on the left hand side.

Construction 2 (OR) Assume two Σ -protocols (P_1, V_1) and (P_2, V_2) for relations R_1 and R_2 , respectively. Assume two efficiently recognisable sets M_1, M_2 of bitstrings. We use the notation from Definition 17 with an index denoting the protocol (e.g., r_2 is the message sent by the verifier V_2).

Let $N \subseteq N_1 \cup N_2$ and let $+$ be some abelian group operation on N . Define (P, V) as follows:

- Prover's input: $((x_1, x_2), (i, w)) \in R_1 \vee_{M_1, M_2} R_2$.
- Verifier's input: (x_1, x_2) .
- Assume $i = 1$, in the case $i = 2$ the prover's code is analogous and the verifier's code identical.
- The prover computes $a_1 \leftarrow P_1(x_1, w)$ and $r_2 \xleftarrow{R} N$ and $(a_2, r_2, s_2) \leftarrow S_2(x_2, r_2)$ where S_2 is the special HVCZK simulator for (P_2, V_2) .
- The prover sends (a_1, a_2) to the verifier.
- The verifier picks $r \xleftarrow{R} N$ and sends r to the prover.
- The prover computes $r_1 := r - r_2$ and $s_1 \leftarrow P_1(r_1)$ and sends (r_1, r_2, s_1, s_2) to the verifier.
- The verifier checks that $x_1 \in M_1$, and $x_2 \in M_2$, and $r_1, r_2 \in N$, and $r = r_1 + r_2$ and that V_1 would accept (a_1, r_1, s_1) and V_2 would accept (a_2, r_2, s_2) .

(Note: We assume that the prover P_1 keeps state between its activations. Hence in the invocation $P_1(r_1)$ which produces the prover's second message when getting the message r_1 , the prover additionally knows x_1, w from its first activation. Same for P_2 .)

Theorem 13 Assume that (P_1, V_2) and (P_2, V_2) are special HVCZK and have special soundness and overwhelming completeness (for relations R_1 and R_2 , respectively). Assume that $M_1 \supseteq L_{R_1}$ and $M_2 \supseteq L_{R_2}$. Assume that for both schemes, the verifier accepts the output of the simulator with overwhelming probability.

Then (P, V) as in Construction 2 is special HVCZK and has special soundness and overwhelming completeness for the relation $R_1 \vee_{M_1, M_2} R_2$ (where all negligible functions in the definition of completeness and HVCZK are negligible in $\min\{|x_1|, |x_2|\}$, not $|(x_1, x_2)|$).

The intuition behind the sets M_1 and M_2 is that they correspond to the set of well-formed inputs. For example, in the Schnorr's Σ -protocol for discrete logarithms (Definition 16), a statement would be well-formed if it is a group element, i.e., $M = G$. Then it is easy to see that the simulator for Schnorr's proof system convinces the verifier for all $x \in M = G$. For most efficient Σ -protocols, there are similar natural sets M of well-formed inputs for which the simulator produces an accepting conversation. Hence Construction 2 can be applied for most efficient Σ -protocols. Furthermore, since

usually an honest prover would rarely try to prove $x_1 \vee x_2$ where one of these is actually not even well-formed, for most applications $R_1 \vee_{M_1, M_2} R_2 := \{((x_1, x_2), (i, w)) : i \in \{1, 2\}, (x_i, w) \in R_i \wedge x_1 \in M_1 \wedge x_2 \in M_2\}$ is as useful as the relation $R_1 \vee R_2 := \{((x_1, x_2), (i, w)) : i \in \{1, 2\}, (x_i, w) \in R_i\}$.

A protocol for $R_1 \wedge R_2$ is constructed in exercise sheet 9.

9 Non-interactive zero-knowledge proofs

For some applications, it is necessary to have zero-knowledge proofs that consist of only a single message. These are called non-interactive zero-knowledge (NIZK) proofs. More exactly, a proof system is called non-interactive if it consists of only a single message from prover to verifier and the verifier does not send any message.

Usually, when analysing NIZK proofs, one works in the CRS-model (as in Definitions 22 and 23. For non-interactive proofs, the definition specialises as follows:

Definition 25 (NICZK proof) *A triple $(P, V, KeyGen)$ of polynomial-time algorithms is called a non-interactive computational zero-knowledge proof in the CRS-model for a relation R if the following holds:*

- *Completeness: There is a negligible function μ such that for all $(x, w) \in R$, we have $\Pr[V(x, crs, \pi) = 1 : crs \leftarrow KeyGen(1^{|x|}), \pi \leftarrow P(x, crs, w)] \geq 1 - \mu(|x|)$.*
- *Soundness: For any (possibly unbounded) algorithm P^* there is a negligible function μ such that for all $x \notin L_R$, we have that $\Pr[V(x, crs, \pi) = 1 : crs \leftarrow KeyGen(1^{|x|}), \pi \leftarrow P^*(crs)] \leq \mu(|x|)$.*
- *Computational zero-knowledge: There is a polynomial-time algorithm S such that*

$$\left\{ (crs, \pi) : crs \leftarrow KeyGen(1^{|x|}), \pi \leftarrow P(x, crs, w) \right\}_{x, (w, z)} \quad \text{and} \quad \left\{ S(x) \right\}_{x, (w, z)}$$

are computationally indistinguishable for all $(x, w) \in R, z \in \{0, 1\}^$.*

As a technical tool, we also consider a different model, the hidden-bits model. This model is not very realistic, but it is useful as an intermediate step in constructing NICZK proofs in the CRS model. In the hidden-bit model, there is a kind of CRS hb , but only the prover is able to read all bits of hb . The prover may decide which bits of hb the verifier is allowed to see.

Definition 26 (NICZK proof in the hidden-bit model) *A triple $(P, V, HBGen)$ of polynomial-time algorithms is called a non-interactive computational zero-knowledge proof in the hidden-bit (HB) model for a relation R if the following holds:*

- *Completeness: There is a negligible function μ such that for all $(x, w) \in R$, we have $\Pr[V(x, \pi, I, hb_I) = 1 : hb \leftarrow HBGen(1^{|x|}), (\pi, I) \leftarrow P(x, hb, w)] \geq 1 - \mu(|x|)$.*
Here I denotes a set of indices of bits in hb , and hb_I is the sequences of all bits hb_i with $i \in I$.

- Soundness: For any (possibly unbounded) algorithm P^* there is a negligible function μ such that for all $x \notin L_R$, we have that $\Pr[V(x, \pi, I, hb_I) = 1 : hb \leftarrow HBGen(1^{|x|}), (\pi, I) \leftarrow P^*(hb)] \geq 1 - \mu(|x|)$.
- Computational zero-knowledge: There is a polynomial-time algorithm S such that $\left\{ (\pi, I, hb_I) : hb \leftarrow HBGen(1^{|x|}), (\pi, I) \leftarrow P(x, hb, w) \right\}_{x,w}$ and $\left\{ S(x) \right\}_{x,w}$ are computationally indistinguishable for all $(x, w) \in R$.

Construction 3 (NICZK in CRS model from NICZK in HB model) Let $(P, V, HBGen)$ be a NICZK proof in the HB model. Let a length-preserving one-way permutation f with hard-core bit b be given. Then we define $(P', V', KeyGen)$ as follows:

- $KeyGen(1^n)$ computes $hb_1, \dots, hb_m := hb \leftarrow HBGen(1^n)$, $s_i \xleftarrow{R} \{s \in \{0, 1\}^n : b(s) = hb_i\}$ for $i = 1, \dots, m$, and $crs := (f(s_1), \dots, f(s_m))$. It returns crs .
- The prover $P'(x, crs, w)$ computes $(crs_1, \dots, crs_m) := crs$, $s_i := f^{-1}(crs_i)$ for $i = 1, \dots, m$, $hb_i := b(s_i)$ for $i = 1, \dots, m$, and $(\pi, I) \leftarrow P(x, hb, w)$. It sets $\pi' := (\pi, I, s_I)$ and outputs π' .
- The verifier $V'(x, crs, \pi')$ computes $(crs_1, \dots, crs_m) := crs$, $(\pi, I, s_I) := \pi'$, and $hb_i := b(s_i)$ for $i \in I$. It checks that $crs_i = f(s_i)$ for all $i \in I$ and that $V(x, \pi, hb_I) = 1$. In this case, it outputs 1.

Theorem 14 If $(P, V, HBGen)$ is a NICZK proof in the HB model, and f is a length-preserving one-way permutation with hard-core bit b , then $(P', V', KeyGen)$ from Construction 3 is a NICZK proof in the CRS model. However, P' is not polynomial time.

The construction can be modified to produce a polynomial-time P' (assuming the existence of trap-door permutations), but this only works if the output of $HBGen$ is a uniformly distributed bitstring.

Furthermore, NICZK proofs in the HB model can be produced for any language in NP using the following result:

Theorem 15 Let $R := \{(G, H) : H \text{ is a Hamiltonian cycle of } G\}$. Then there exists a NICZK proof $(P, V, HBGen)$ in the HB model for R with perfect completeness, perfect soundness, and perfect zero-knowledge. However, the output of $HBGen$ is not uniformly distributed on bitstrings.

The construction can be modified to yield a scheme where $HBGen$ is uniformly distributed on bitstrings, but the resulting scheme does not have perfect soundness any more. (And is much more inefficient.)

Corollary 4 Assume that one-way permutations exist. For any NP-relation R there exists a NICZK proof system in the CRS model (with unbounded prover).

And using enhanced trapdoor permutations, there exists a NICZK proof system in the CRS model with polynomial-time prover.

10 Chosen-ciphertext secure encryption

Definition 27 (Encryption scheme) A (public key) encryption scheme is a triple (K, E, D) of polynomial-time algorithms such that for all $m \in \{0, 1\}^*$, $n \in \mathbb{N}$, we have

$$\Pr[m = m' : (pk, sk) \leftarrow K(1^n), c \leftarrow E(pk, m), m' \leftarrow D(sk, c)] = 1.$$

Definition 28 (IND-CPA) An encryption scheme (K, E, D) is IND-CPA secure if for all non-uniform polynomial-time algorithms A , we have that

$$\Pr[b = b' : (pk, sk) \leftarrow K(1^n), (m_0, m_1) \leftarrow A(1^n, pk), b \xleftarrow{R} \{0, 1\}, c^* \leftarrow E(pk, m_b), b' \leftarrow A(1^n, c^*)]$$

is $\leq \frac{1}{2} + \mu$ for some negligible μ . (Where A is allowed to keep state between the invocations and A must output (m_0, m_1) with $|m_0| = |m_1|$.)

Definition 29 (IND-CCA2) An encryption scheme (K, E, D) is IND-CCA2 secure if for all non-uniform polynomial-time oracle machines A , we have that

$$\Pr[b = b' : (pk, sk) \leftarrow K(1^n), b \xleftarrow{R} \{0, 1\}, (m_0, m_1) \leftarrow A^{D(sk, \cdot)}(1^n, pk), \\ c^* \leftarrow E(pk, m_b), b' \leftarrow A^{D(sk, \cdot)}(1^n, c^*)]$$

is $\leq \frac{1}{2} + \mu$ for some negligible μ . (Where A is allowed to keep state between the invocations and A must output (m_0, m_1) with $|m_0| = |m_1|$ and A is not allowed to query $D(sk, \cdot)$ with c^* .) By $D(sk, \cdot)$ we denote an oracle that upon query x computes and returns $D(sk, x)$.

Definition 30 (One-time signature scheme) A strongly unforgeable one-time signature scheme is a triple of algorithms $(KS, Sig, Verify)$ such that the following properties hold:

- Perfect correctness: For all $m \in \{0, 1\}^*$, $n \in \mathbb{N}$, we have

$$\Pr[Verify(vk, \sigma, m) = 1 : (vk, sigk) \leftarrow KS(1^n), \sigma \leftarrow Sig(sigk, m)] = 1.$$

- Strong one-time unforgeability: For all non-uniform polynomial-time A , we have that

$$\Pr[Verify(vk, \sigma', m') = 1 \text{ and } (m', \sigma') \neq (m, \sigma) : \\ (vk, sigk) \leftarrow KS(1^n), m \leftarrow A(1^n, vk), \sigma \leftarrow Sig(sigk, m), (m', \sigma') \leftarrow A(\sigma)]$$

is negligible in n . (Here A is allowed to keep state between invocations.)²¹

Definition 31 (Unbounded adaptive NICZK argument) An adaptive NICZK argument with perfect completeness in the CRS model for a relation R is a triple $(P, V, KeyGen)$ of polynomial-time algorithms satisfying the following properties:

²¹We get a weaker property called one-time unforgeability if we write $m' \neq m$ instead of $(m', \sigma') \neq (m, \sigma)$.

- Perfect completeness: For all $(x, w) \in R$ we have $\Pr[V(x, crs, \pi) = 1 : crs \leftarrow KeyGen(1^n), \pi \leftarrow P(x, crs, w)] = 1$.
- Adaptive computational soundness: For any non-uniform polynomial-time algorithm P^* there is a negligible function μ such that $\Pr[V(x, crs, \pi) = 1 \text{ and } x \notin L_R : crs \leftarrow KeyGen(1^n), (x, \pi) \leftarrow P^*(crs)] \leq \mu(n)$.
- Adaptive computational zero-knowledge: There is a polynomial-time algorithm S such that for any A polynomial in its first input, we have that

$$\left\{ (x, crs, \pi) : crs \leftarrow KeyGen(1^n), (x, w) \leftarrow A(1^n, crs, z), \pi \leftarrow P(x, crs, w) \right\}_{1^n, z}$$

and

$$\left\{ (x, crs, \pi) : crs \leftarrow S(1^n), (x, w) \leftarrow A(1^n, crs, z), \pi \leftarrow S(x) \right\}_{1^n, z}$$

are computationally indistinguishable for all $n \in \mathbb{N}$, $z \in \{0, 1\}^*$. (Here A is required to output $(x, w) \in R$, and S may keep state between invocations.)

The word *adaptive* denotes the fact, that the statement x is chosen in dependence of the CRS. The word *unbounded* denotes the fact that for a given CRS, one can prove statements x of arbitrary length. (In comparison, in Definition 25, the algorithm *KeyGen* gets the length of x as input when constructing the CRS.)

Such proofs can be constructed roughly as follows: Start with the proof system for HC (with efficient prover). This proof system already has adaptive NICZK. By parallel amplification, amplify the soundness to $2^{-(|x|)^2}$. The fact that the prover may choose x on its own may increase the soundness bound only by a factor of $2^{|x|}$, hence the resulting system still has adaptive soundness $2^{-|x|}$. From the resulting scheme, construct a multi-theorem NICZK proof using the construction from exercise sheet 10, problem 2. To make the proof system unbounded, observe that we can split any statement of arbitrary length into many smaller statements as follows: Let C be a circuit that checks whether w is a witness. Then commit to the bits of w and to all intermediate results in the computation of $C(w)$ (so for each gate, one commitment). Then, for each gate, make a proof that the value in the commitment is the AND or the OR (depending on the gate) of the values in the commitments corresponding to the inputs of the gate. All commitments and all these proofs together form a proof of the original statement. Hence we have constructed a proof system to prove statements x of unbounded length. (Not very efficient, though.)

Construction 4 (Dolev-Dwork-Naor [DDN00]²²) Let (K, E, D) be an IND-CPA secure encryption scheme. Let $(KS, Sig, Verify)$ be a strongly unforgeable one-time signature scheme. Let ℓ denote the length of a verification key. Let $(KeyGen, P, V)$ be an unbounded adaptive NICZK argument with perfect completeness.

Then we construct the encryption scheme (K', E', D') as follows:

²²Actually, Dolev-Dwork-Naor is constructed slightly differently: They additionally use a hash function H , and instead of using the bits vk_i to select public keys, they use the bits h_i where $h = H(vk)$.

- *Key Generation.* Upon input 1^n , K' performs the following steps:
 - $crs \leftarrow \text{KeyGen}(1^n)$.
 - $(pk_i^j, sk_i^j) \leftarrow K(1^n)$ for $i = 1, \dots, \ell$, $j = 0, 1$.
 - $pk := (crs, pk_1^0, pk_1^1, \dots, pk_\ell^0, pk_\ell^1)$.
 - $sk := (crs, pk_1^0, pk_1^1, \dots, pk_\ell^0, pk_\ell^1, sk_1^0, sk_1^1, \dots, sk_\ell^0, sk_\ell^1)$.
 - Return (pk, sk) .
- *Encryption:* Upon input (pk, m) , E' performs the following steps:
 - $(crs, pk_1^0, pk_1^1, \dots, pk_\ell^0, pk_\ell^1) := pk$.
 - $(vk, sigk) \leftarrow KS(1^n)$.
 - $c_i \leftarrow E(pk_i^{vk_i}, m)$ for $i = 1, \dots, \ell$. Let r_i denote the randomness used in the i -th invocation of E .
 - $\pi \leftarrow P((c_1, \dots, c_\ell, pk_1^{vk_1}, \dots, pk_\ell^{vk_\ell}), (m, r_1, \dots, r_\ell))$.
 - $\sigma \leftarrow \text{Sig}(sigk, (c_1, \dots, c_\ell, \pi))$.
 - Return $c := (c_1, \dots, c_\ell, \pi, \sigma, vk)$.
- *Decryption:* Upon input (sk, c) , D' performs the following steps:
 - $(crs, pk_1^0, pk_1^1, \dots, pk_\ell^0, pk_\ell^1, sk_1^0, sk_1^1, \dots, sk_\ell^0, sk_\ell^1) := sk$.
 - $(c_1, \dots, c_\ell, \pi, \sigma, vk) := c$.
 - Check whether $\text{Verify}(vk, \sigma, (c_1, \dots, c_\ell, \pi)) = 1$.
 - Check whether $V((c_1, \dots, c_\ell, pk_1^{vk_1}, \dots, pk_\ell^{vk_\ell})) = 1$.
 - If a check fails, return \perp .
 - Return $m := D(sk_1^{vk_1}, c_1)$.

Theorem 16 *The scheme (K', E', D') from Construction 4 is an IND-CCA2 secure encryption scheme.*

References

- [Dam00] Ivan Damgård. Efficient concurrent zero-knowledge in the auxiliary string model. In *EUROCRYPT 2000*, volume 1807 of *LNCS*. Springer, 2000. Online available at <http://www.iacr.org/archive/eurocrypt2000/1807/18070424-new.pdf>.
- [DDN00] Danny Dolev, Cynthia Dwork, and Moni Naor. Nonmalleable cryptography. *SIAM Journal on Computing*, 30(2):391–437, 2000. Online available at <http://www.wisdom.weizmann.ac.il/%7EEnaor/PAPERS/nmc.ps>.
- [Gol01] Oded Goldreich. *Foundations of Cryptography – Volume 1 (Basic Tools)*. Cambridge University Press, August 2001. Previous version online available at <http://www.wisdom.weizmann.ac.il/~oded/frag.html>.

Index

- Σ -protocol, 13
- (interactive) proof system, 2
- 3-colouring
 - graph, 8
- adaptive NICZK, 20
- adaptive non-interactive argument, 20
- amplification, 6
- argument, 15
- argument of knowledge, 15
- auxiliary input, 6
- CCA2 security, 20
- commitment, 8
 - equivocal, 14
 - trapdoor, 14
- completeness, 2
 - perfect, 3
- composition theorem
 - sequential, 6, 11
- computational soundness, 15
- computational indistinguishability, 9
- computational zero-knowledge
 - in CRS model, 15
 - non-interactive, 18
- computational zero-knowledge, 10
- computationally sound proof, 15
- CPA security, 20
- CRS model, 15
- discrete logarithm, 12
- encryption scheme, 20
- equivocal commitment, 14
- extractability, 12
- graph-3-colouring, 8
- hidden-bit model, 18
- honest-verifier zero-knowledge, 13, 14
- HVCZK, 13, 14
- IND-CCA2 security, 20
- IND-CPA security, 20
- indistinguishability
 - computational, 9
 - statistical, 9
- interactive machine, 2
- knowledge
 - proof of, 11
- knowledge error, 11
- logarithm
 - discrete, 12
- negligible, 1
- NICZK, 18
- NIZK, 18
- non-interactive zero-knowledge, 18
- non-negligible, 1
- nondegeneracy
 - statistical distance, 3
- nonnegativity
 - statistical distance, 3
- noticeable, 1
- NP-complete relation, 7
- NP-relation, 7
- one-time signatures, 20
- oracle access
 - with rewinding, 11
- overwhelming, 1
- perfect completeness, 3
- poly-balanced, 7
- proof
 - computationally sound, 15
 - non-interactive, 18
- proof of knowledge, 11
- public-key encryption, 20
- random tape, 2
- relation
 - NP-complete, 7

- rewinding, 11
- Schnorr's proof system, 12
- sequential composition theorem, 6
- Σ -protocol, 13
- soundness, 2
 - computational, 15
 - special, 13
- special soundness, 13
- statistical distance
 - nondegeneracy, 3
 - triangle inequality, 3
- statistical indistinguishability, 9
- statistical distance, 3
 - nonnegativity, 3
 - properties, 3
 - symmetry, 3
- statistical zero-knowledge, 4, 6
- strongly unforgeable signatures, 20
- symmetry
 - statistical distance, 3
- three-colouring
 - graph, 8
- trapdoor commitment, 14
- triangle inequality
 - statistical distance, 3
- unbounded NICZK, 20
- unforgeable signatures, 20
- validity, 11, 15
- zero-knowledge
 - computational, 10
 - honest-verifier, 13, 14
 - non-interactive, 18
 - statistical, 4, 6