Non-Interactive Zero-Knowledge: A survey

Lauri Tarkkala

October 30, 2001

1 Introduction

The concept of zero-knowledge provides a framework for designing and analysing cryptographic protocols that should be highly resistant to attack from an adversary. [2]

The practical applications, such as e-mail, which could benefit from strong cryptographic protocols for authentication, confidentiality, integrity and so on unfortunately do not allow for the round-complexity of interactive protocols. [6] [5]

The research and development of non-interactive zero-knowledge protocols provides tools and techniques for implementing zero-knowledge primitives for non-interactive protocols or improving upon the non zero-knowledge ones.

The existence of non zero-knowledge protocols was originally shown to rely the Quadratic Residuosity Assumption (QRA). A theorem was later established which shows that if one-way permutations exist, then so do non-interactive zero-knowledge protocols for all NP-complete languages.

A non-interactive zero knowledge protocol can be based on any one-way permutation or trapdoor permutation. The practical protocols, which unfortunately must accomodate a prover operating under some limitations are often based on trapdoor permutations.

These protocols can be used to build public-key cryptosystems resistant to chosen ciphertext attack or digital signature schemes. This survey shows some parts of the theoretical framework underlying non-interactive zero-knowledge protocols and highlights the most important results.

2 Non-Interactive Zero-Knowledge

A non-interactive proof system is closely related to the interactive proof systems shown in previous surveys. [1] The obvious additional restriction is that the prover P outputs a proof which the verifier V must accept or reject without querying the prover P.

A relaxation is added, and P and V are allowed to share a string of random bits, which is of polynomial length in the input and uniformly distributed. [2]

Definition 2.1 A pair of probabilistic algorithms (V, P) is a non-interactive proof system for a language $L \subseteq \Sigma^*$ if the following conditions are met:

- V polynomial-time.
- *Completeness*
  $\forall x \in L : \Pr(V(x, R, P(x, R)) = 1) \geq 1 - \varepsilon$
- *Soundness*
  $\forall x \notin L : \forall B : \Pr(V(x, R, B(x, R)) = 1) \leq \varepsilon.$
- The common reference string $R \in \{0,1\}^{poly(|x|)}$ is polynomial in the length of the problem instance.

The error probability ε of the algorithm must ne non-negligible. A common value used is $\varepsilon = 1/3$.

Note that because the protocol is non-interactive, a reduction in the error probability performed by using repetitions of the protocol means that several (different) copies of the random reference string must be used, or the output of the prover with a fixed input x and reference stream R must be distributed over a sufficiently large set.

The latter case would imply that for each problem instance and common reference string pair there could be expected to exist a sufficiently large set of valid proofs (s.t. that the soundness criteria is satisfied, hence the amount of correct proofs must be a fraction of all possible proof strings). This is cumbersome compared to the simplicity of using several different reference strings and having the prover generate several proofs for each (x, R) pair. [6]

Definition 2.2 Let U_m be a random variable uniformly distributed over $\{0,1\}^m$.

Let $M : L \to L \times U_{p(|x|)} \times P(x, U_{p(|x|)})$ be a polynomial time probabilistic algorithm which maps a problem instance to proofs of the form $(x, R, P(x, R))$.

Let p be a polynomial.

Let $\alpha = \{M(x)\}_{x \in L}$ be a sequence of functions.

Let $\beta = \{(x, U_{p(|x|)}, P(x, U_{p(|x|)}))\}$ be a sequence of functions which maps problem instances to proofs, such that the random string is uniformly chosen.

The proof system (V, P) is zero-knowledge if for a prover P there exists a machine M and a polynomial p s.t. β and α are polynomially indistinguishable

Informally a non-interactive proof system is zero-knowledge if there exists a probabilistic polynomial-time machine M which for all problem instances produces a distribution of proofs which is computationally indistinguishable from the proofs produced by P when R is uniformly distributed.

This definition does not consider the case where the problem instance is chosen in an adversarial manner after R has been fixed. This more challenging case is called "adaptive non-interactive zero-knowledge" and is considered further on in this survey.

The definitions above also consider the case where only one problem instance is proven with respect to a common reference string of polynomial length in the input. The zero-knowledge property, the completeness or soundness property are not guaranteed directly by the above definitions to hold if e.g. an amount of assertions linear with respect to the length of the common reference string R are proven utilising the same R.

3 NP $\subseteq$ NIZK

NIZK is the class of languages for which there exist non-interactive zero-knowledge proof systems, as defined above.

Several proofs for **NP** $\subseteq$ **NIZK** have been shown in the literature. The early proofs by Blum, Feldman and Micali and De-Santis Micali and Persiano and Boyar and Peralta relied on the number-theoretic assumption that deciding quadratic residuosity for an integer residue modulo a Blum Integer n with Jacobi symbol 1 modulo n is an intractable problem. [6] [4]

Some other protocols relied on a pre-processing step, allowing the prover and verifier to perform protocol rounds (independent of the assertion to be proven), before an actual proof was to be created.

Protocols for **NP**-complete problems based complexity-theoretic assumptions were first intrduced by Feige, Lapidot and Shamir. The prover in the protocol generated a trapdoor permutation, which was then contained in the proof sent to the verifier. [4] [3]

Bellare and Yung later showed that if the proover could construct a suitable function in such a protocol and send it in the proof instead of a permutation, the prover may be able to cheat. Simplifying, such a cheating function would map several inputs to a suitable output. This would allow the cheating prover to "shape" the distribution of the protocol from that implied by the assumption of the verifier that the "bad function" was actually a permutation. [3]

This problem was solved by Bellare and Yung for a construction of a non-interactive zero-knowledge protocol for convincing a verifier that a mapping $f : \{0,1\}^n \to \{0,1\}^n$ has more than one pre-image for at most $\varepsilon * 2^n$ points in the range of the map.

The protocol sketched out below relies on a so-called hidden bit construction, and is probably the most simple of all the protocols in the literature for showing that **NP** $\subseteq$ **NIZK**.

3.1 Hidden-Bits Proof Systems

This briefly describes the hidden-bits proof system, which attempts to provide an easier framework for building non-interactive zero-knowledge protocols. [2]

Definition 3.1 A pair of probabilistic algorithms is a hidden-bits proof system for L if the following conditions hold.

- V is probabilistic polynomial time.
- *Completeness*
  $\forall x \in L : \Pr(V(x, R_I, I, \Pi) = 1) \geq 1 - \varepsilon$ when $(I, \Pi) = P(x, R)$.
- *Soundness*
  $\forall x \notin L : \forall B : \Pr(V(x, R_I, I, \Pi) = 1) \leq \varepsilon$ when $(I, \Pi) = B(x, R)$.
- The common reference string $R \in \{0,1\}^{poly(|x|)}$ is polynomial in the length of the problem instance s.t. $m = poly(|x|)$.
- R_I is the sub-string of R at positions $I \subseteq \{1, 2, ..., poly(|x|)$.

Π is called the certificate and I is called the set of revealed bits.

The definition above is used as the framework for constructing a protocol for deciding the hamilton circuit problem and then using a construction to transform that protocol into a non-interactive zero-knowledge protocol.

A hidden-bits proof system is zero-knowledge if there exists a probabilistic polynomial time algorithm, which for all problem instances produces a distribution of proofs which is computationally indistinguishable from the proofs produced by P when R is uniformly distributed.

3.2 Hidden-Bits Construction

This section presents the hidden-bits construction, which can be used to construct a non-interactive zero-knowledge protocol from a hidden-bits protocol. [2]

Let (V, P) be a hidden-bits proof system for a language L and let $f : \{0,1\}^* \to \{0,1\}^*$ and $b : \{0,1\}^* \to \{0,1\}$ be polynomial-time algorithms. Assume that f is one-to-one and length-preserving.

Let $m = poly(|x|)$ be the length of the common reference string for input x of the constructed non-interactive zero-knowledge protocol.

Define the non-interactive proof-system (V', P') as follows:

- Let x be the input (problem instance).
- Let $s = (s_1, ..., s_m)$, $s_i \in \{0,1\}^n$ be the common reference string.

The prover P' now simply computes $r_i = b(f^{-1}(s_i))$ for all m sub-strings of the common reference string, producing a m bit string to be used as the common reference string to the hidden-bits protocol prover P.

P' then computes $(I, \Pi) = P(x, r_1, ..., r_m)$.

The proof is then presented by P' as (I, Π, p_I) where $p_I = (f^{-1}(s_{i_1}), ..., f^{-1}(s_{i_t}))$ for $I = (i_1, ..., i_t)$.

The verifier V' then functions as follows.

- Let $r_i = b(p_i)$ for $i = 1, ..., t$, let $r = r_1, ..., r_t$.
- If $s_{i_j} \neq f(p_j)$ for any $i_j \in I$ then V' rejects.
- If V accepts the proof (x, r, I, Π) then V' accepts.
- If V rejects the proof (x, r, I, Π) then V' rejects

Theorem 3.2 (V', P') *is a non-interactive proof-system for* L *if* $\Pr(b(U_n) = 1) = 1/2$*. If* P *is zero-knowledge,* f *is a one-way permutation and* b *is a hard-core of* f*, then* P' *is zero-knowledge.*

A sketch of the proof to the above theorem is provided in [2] and is not repeated here.

An actual non-interactive zero-knowledge protocol based on the hidden-bits construction unfortunately requires the ability to invert a oneway permutation.

A practical implementation must therefore must rely on the existence of a trapdoor oneway function. Note that in this case the oneway function must be provided to the verifier, and the verifier must then check that the oneway function is actually a permutation.

3.3 Hidden-Bits Model for HC

This section refers a zero-knowledge hidden-bits protocol for Graph Hamiltonicity.

A permutation matrix is a matrix where each row and column contain a single entry of 1. A Hamiltonian matrix is a permutation matrix that as an adjacency matrix defines a graph containing a single Hamiltonian cycle.

Let $G = (V, E)$ s.t. $n = |V|$ be the common input, where G is the graph to be tested for Hamiltonicity.

Let M be a $n^3 \times n^3$ matrix s.t. for each element $a_{j,i}$ $\Pr(a_{j,i} = 1) = n^{-5}$ and $a_{j,i} \in \{0, 1\}$.

M is called useful if it contains a $n \times n$ Hamiltonian sub-matrix and all other entries in M are 0.

Let C be a Hamiltonian cycle in G, assuming it exists.

If M is useful, then prover P acts as follows.

- Let H denote the Hamiltonian sub-matrix and C_H the cycle defined by H.
- P reveals all n^4 entries in M that are not in H.
- P finds one-to-one mappings π_1 and π_2 s.t. π_1 is from V to the rows of H and π_2 is from V to the columns of H and the edges of C are mapped to the 1-entries of H.
- P reveals to V the "hidden bits" in H which do not correspond to edges in G.
- P outputs (π_1, π_2) as a certificate.

If M is not useful, then P reveals all the "hidden bits" in M and no certificate is provided.

The verifier acts as follows.

- If the prover has not revealed all entries in M, then the verifier checks that all entries in M that do not have pre-images under (π_1, π_2) are zero and have been revealed. This can be done by simply permuting all nodes in $V \times V \setminus E$ and checking the appropriate element in M.
- If the prover has revealed all bits in M, then the verifier accepts iff M is useful.

The protocol above is a zero-knowledge hidden bits protocol for Graph Hamiltonicity. A proof is sketched in [2] not referred here.

This results in the following theorems together with the hidden-bits construction shown above.

Theorem 3.3 *If there exists one-way permutations, then* $\mathbf{NP} \subseteq \mathbf{NIZK}$.

Theorem 3.4 *If there exist families of trapdoor permutations, then* $\mathbf{NP} \subseteq \mathbf{NIZK}$ *and for all languages* $L \in NP$ *there exists a non-interactive zero-knowledge proof system* (V, P) *such that* P *can be computed in probabilistic polynomial time if it receives the appropriate* **NP***-witness as input.*

4 Unbounded NIZK

As mentioned earlier, the initial definition of non-interactive zero-knowledge does not account for proofs of multiple different assertions using the same common reference string. The definition of unbounded non-interactive proof systems corrects this.

A non-interactive proof system (V, P) is called unboundedly zero-knowledge if for every polynomial p there exist a probabilistic polynomial-time algorithm M such that a sequence of length $p(n)$ of proofs generated for arbitrary inputs $x_1, ..., x_{p(n)}$ by P when the common reference string is chosen uniformly over $\{0, 1\}^n$ is computationally indistinguishable from the distribution generated by M when it receives $x_1, ..., x_{p(n)}$ as input.

The length of each problem instance x_i is bounded above by n^{ε}. A **NP**-assertion can be split into several assertions bounded above by n^{ε}, and hence the treatment is still general. ε is called the fundamental constant of the proof system.

Any non-interactive zero-knowledge protocol, which operates using a **NP**-witness as additional input, can be transformed into satisfying the criteria unboundedness above. This implies the following result, together with similar results cited earlier.

Theorem 4.1 *If there exist families of trapdoor permutations, each language in* **NP** *has an unboundedly zero-knowledge non-interactive proof system. The prover can be implemented by a probabilistic polynomial time machine, that receives the appropriate* **NP***-witness as input.*

5 Adaptive NIZK

The initial definition of non-interactive zero-knowledge does not take into account an adversary, which chooses the problem instances depending on the common reference string. The definition of adaptive zero-knowledge corrects this.

Let M_1 and M_2 be probabilistic polynomial-time algorithms and letΞ be a function which maps a common reference string to a string in $x \in L$.

$M_1 : 1^* \rightarrow \{0,1\}^*\{0,1\}^*$ generates the common reference and an auxiliary string s of length n specified by a unary input string.

M_2 generates a proof for $\Xi(r) \in L$ using s and $\Xi(r)$ as input. Denote by $M^{\Xi}(1^n)$ the behaviour of M_1, M_2 and Ξ combined in the random generation of a common reference string, problem instance and corresponding proof.

A proof system (V, P) for language L is adaptively zero-knowledge if for all functions Ξ which map the common reference string r to a string $x \in L$ there exist machines M_1 and M_2 s.t. the distribution of the sequences $\{M^{\Xi}(1^n)\}_{n \in N}$ and $\{((U_n, \Xi(U_n), P(\Xi(U_n), U_n))\}_{n \in N}$ are computationally indistinguishable.

U_n denotes again a uniformly chosen random variable over $\{0,1\}^{poly(n)}$.

A construction for modifying a non-interactive zero-knowledge protocol into an adaptively zero-knowledge protocol is not yet known, but it is stated that assuming the existence of one-way permutations, each language in **NP** has a non-interactive proof system that is adaptively zero-knowledge. [2]

Goldreich also states that if trapdoor permutations exist, then for all **NP**-languages there exist non-interactive proof systems which are both adaptively zero-knowledge and unboundedly zero-knowledge. [2]

References

[1] P. Kaski. T-79.511 Special Course on Cryptology / Zero Knowledge: Rudiments of Interactive Proof Systems. Available at `http://www.tcs.hut.fi/Teaching/T-79.511/hutzk1.ps.gz`

[2] O. Goldreich. *Foundations of Cryptography: Basic Tools.* Cambridge University Press, Cambridge, England, 2001.

[3] M. Bellare and M. Yung. Certifying Permutations: Non-Interactive Zero-Knowledge Based on any Trapdoor Permutation. *Journal of Cryptology* **9** (1996) no. 1 p. 149-166.

[4] J. Boyar and R. Peralta. Efficient Non-Interactive Zero-Knowledge Proofs of Circuit Satisfiability. IMADA preprint PP-1994-01, University of Southern Denmark, 1994.

[5] M. Naor and M. Yung. Public-key Cryptosystems Provably Secure Against Ciphertext Attacks. Proc. 22th ACM STOC, p. 427–437, ACM, New York, 1990.

[6] M. Blum, A. Santis, S. Micali and G. Persiano. Non-Interactive Zero-Knowledge. *SIAM Journal on Computing* **20** (1991) no. 6, pp 1084–1118.