HELLO WORLD

Zero-Knowledge Proofs

BY MARINKA ZITNIK

A zero-knowledge proof allows one person to convince another person of some statement without revealing any information about the proof other than the fact that the statement is indeed true. Zero-knowledge proofs are of practical and theoretical interests in cryptography and mathematics. They achieve a seemingly contradictory goal of proving a statement without revealing it. We will describe the interactive proof systems and some implications that zero-knowledge proofs have on the complexity theory. We will then conclude with an application of zero-knowledge proofs in cryptography, the Fiat-Shamir identification protocol, which is the basis of current zero-knowledge entity authentication schemes.

Definition 1: Definitions of parties participating in Fiat-Shamir identification protocol in Python.

```
from fractions import gcd
from numpy.random import random_integers

class Prover():
    def __init__(self, tc, s):
        assert gcd(s, tc.n) == 1, ‘Secret s and n are not coprime.’
        self.n = tc.n
        self.__s = s
        tc.set_up((self.__s**2)%self.n)

    def generate(self):
        self.r = random_integers(1, self.n-1)
        x = (self.r**2)%self.n
        return x

    def response(self, e):
        y = (self.r*self.__s)%self.n if e else self.r
        return y

class Verifier():
    def __init__(self, tc):
        self.tc = tc

    def set_up(self, x):
        self.x = x

    def challenge(self):
        self.e = random_integers(0, 1)
        return self.e

    def verify(self, y):
        test = (self.x*self.tc.v**self.e)%self.tc.n
        if y==0 or (y**2)%self.tc.n != test:
            return False
        return True

class TrustedCenter():
    def __init__(self, n):
        self.n = n

    def set_up(self, v):
        self.v = v
```

Interactive Proof Systems

We first give an overview of an interactive proof system. There are two participants in the system, Peggy and Victor. Peggy is the "*prover*" and Victor is the "*verifier*." Peggy knows a fact and she wishes to communicate to Victor she knows it without revealing it. We can think of Peggy and Victor as being probabilistic algorithms that communicate to each other through a communication channel. Initially, they both possess some input object (i.e. a graph representation). The objective is for Peggy to convince Victor that this object has a specified property, i.e. that it is a yes-instance of a particular decision problem. For instance, we could ask if the given graph is isomorphic to another graph [1] or if a given graph is 3-colorable [1].

The interactive proof is a challenge and response protocol that consists of a number of iterations. In each iteration Peggy and Victor do the following: (1) Victor challenges Peggy with a problem instance, (2) Peggy performs some private computation, and (3) she sends a response to Victor. At the end of the proof, Victor either accepts or rejects, depending on whether or not Peggy successfully replies to all of Victor's challenges. Interactive proof systems have to be sound and complete [1]. A proof is complete if honest Victor will always be convinced of a true statement by honest Peggy. It is sound, if cheating Peggy can convince honest Victor that the same false statement is actually true with only a small probability.

A zero-knowledge proof is an interesting type of an interactive proof.

This is one in which Victor, at the end of the proof, still has no idea of how to prove by himself that an object has a property of interest. Readers can find a formal definition of the zero-knowledge strategy in *Cryptography: Theory and Practice* and "Definitions and Properties of Zero-knowledge Proof Systems" by Goldreich and Oren [1, 2].

Fiat-Shamir Identification Protocol
Zero-knowledge proofs in cryptography have natural applications for entity authentication. We assume Peggy possesses some secret *s* that only she can know. She proves to Victor she is indeed Peggy by proving she possesses that secret. Obviously she wants to do so without revealing the secret to any eavesdropper. The Fiat-Shamir identification protocol [3] serves as the basis of modern zero-knowledge identification protocols, such as Feige-Fiat-Shamir and Guillou-Quisquater schemes.

Three parties (see Definition 1) participate in the protocol, which consists of two phases: initialization and identification (see Definition 2). In initialization a trusted center selects two primes p and q, keeps them secret and publishes the $n=pq$. Then Peggy selects a secret number s that is coprime to n, computes $v=s^2 \mod n$ and registers v with trusted center as her public key. The identification phase is repeated t times and if Victor successfully completes all t iterations, he accepts. In each iteration, Peggy chooses a random r and sends $x=r^2 \mod n$ to Victor. Then Victor randomly selects a bit b and sends it to Peggy. She privately computes $y=r$ (if b=0) or $y=rs$ (if b=1) and sends y to Victor. Finally, Victor rejects if y=0 or if $y^2 \not\equiv x\, v^b \pmod n$.

The Fiat-Shamir protocol is complete because honest Peggy can always correctly provide Victor with y based on bit b that he selected. Therefore, honest Victor will successfully complete all t iterations and will accept with probability 1. If Peggy (or an impostor) does not possess the secret *s*, then she can provide only a random guess of *y*=*r* or *y*=*rs*. Honest Victor will reject with probability ½ in every iteration. That implies an overall probability of 2^{-t} that cheating Peggy will not be caught and as a result the Fiat-Shamir protocol is sound. The Fiat-Shamir scheme also upholds the property of zero-knowledge. The only information revealed in each round is the *x* and *y*. Such pairs (*x*,*y*) could be simulated by choosing *y* randomly and then computing the corresponding *x*. These pairs are computationally indistinguishable from pairs generated by the protocol.

Definition 1 is a straight-forward implementation of the described Fiat-Shamir identification protocol. Let see an example with *p*=7 and *q*=5. Then *n*=35 and *n* is published to a trusted center. Let assume Peggy secretly chooses *s*=16, which is coprime to 35. She publishes *v*=11 to the trusted center. Victor requires 10 successful rounds of the protocol in order for him to accept (see Definition 3).

Definition 2: Initialization and identification phases of Fiat-Shamir identification protocol in Python.

```
def fiat_shamir_initialization(n, s):
    tc = TrustedCenter(n)
    peggy = Prover(tc, s)
    return tc, peggy

def fiat_shamir_identification(t, tc, peggy):
    victor = Verifier(tc)
    for _ in xrange(t):
        victor.set_up(peggy.generate())
        c = victor.verify(peggy.response(victor.challenge()))
        if not c:
            print ‘Reject’
            return
    print ‘Accept’
```

Definition 3: An example run of the Fiat-Shamir identification protocol. Suppose the trusted center selects an RSA-like modulus *n*=35, Peggy secretly chooses *s*=16, and Victor requires *t*=10 successful iterations of the protocol.

```
>>> tc, peggy = fiat_shamir_initialization(35, 16)
>>> fiat_shamir_identification(10, tc, peggy)
Accept
```

Zero-Knowledge Proofs and NP Complexity Class

Zero-knowledge proofs exist for decision problems, such as graph isomorphism, 3-colorability, quadratic residuosity, and non-residuosity. Readers would now ask, for which problems can we design zero-knowledge proofs. Powerful and general result exists [4] that informally say that any language for which membership can be efficiently verified can be proved in zero-knowledge. Zero-knowledge proofs exist for all problems in NP, provided that one-way functions exist. That result is utilized for the design of cryptographic protocols, because it enforces parties to behave according to predetermined standards.

Conclusion

Zero-knowledge proofs have some fascinating applications. We might use them to enforce honest behavior. For instance, parties in an interactive game could prove they are not cheating. At the outset of the game, parties commit to the secret inputs and random coins of the prescribed tools they are supposed to use. They then carry out the game procedures and with each output message they prove to each other in zero-knowledge that the message was honestly obtained under the committed inputs and random coins. Properties of zero-knowledge systems guarantee us that participants have to act honestly in order to be able to provide a valid proof (i.e. soundness) and the proofs cannot compromise the privacy of their secret inputs (i.e. zero-knowledge).

Zero-knowledge systems are useful to assure deniability and prevent unwanted transfer of information. Suppose Alice wants to prove her classmate Bob that she did her essay homework. One way to do this is for Alice to show her homework to Bob. However, what if Bob is ignorant and wants to cheat by copying Alice’s essay? The problem is an essay identifying Alice as an author is transferable. Instead, Alice should prove to Bob using zero-knowledge that she did the work. This is an NP-statement since the work is a valid witness, which Alice has in her possession. Bob will believe the proof, but he will not be able to convincingly transfer the transcript of that proof to anybody else. For all we know, Bob could have created the encoded transcript of the homework on his own by running a simulator. In other words, Alice’s proof is deniable, in that she can plausibly claim she was not responsible for producing it.

References

[1] Stinson, D. R. *Cryptography: Theory and Practice.* Chapman & Hall, Boca Raton, FL, 2005.

[2] Oded Goldreich and Yair Oren. Definitions and properties of zero-knowledge proof systems. *Journal of Cryptology*, 7, 1 (1994), 1-32.

[3] Fiat, A. and Shamir, A. How to prove yourself: practical solutions to identification and signature problems. *Advances in Cryptology-Crypto’86*, (1987), 186-194.

[4] Goldreich, O., Micali, S., and Wigderson, A. Proofs that yield nothing but their validity or all languages in NP have zero-knowledge proof systems. *Journal of the ACM*, 38, 3 (1991), 690-728.