# Resettable Zero Knowledge

**Johan Wallén**

**November 14, 2001**

# The basic idea

A **RZK** protocol [Canetti, Goldreich, Goldwasser, Micali 2000], is a protocol that remains zero knowledge, even if the adversary can interact with the prover **many times**, each time **resetting** the prover to its **initial state**.

The initial state of the prover especially includes its **random string**.

Realistic scenario: Consider a simple device (e.g. smart card) without a power supply or writable non-volatile memory.

None of the previously proposed protocols are secure in this model.

# An example

Consider the standard zero knowledge protocol for GRAPH 3-COLORING.

1. The prover commits to $t$ random relabelings of the coloring.
2. The verifier picks $t$ random edges $(u_i, v_i)$.
3. The prover reveals the colors of $u_i$, $v_i$ relative to the $i$th relabeling.
4. The verifier verifies that $u_i$ and $v_i$ have different colors.

This is clearly insecure in the resettable model.

# An example (cont)

When the prover is reset to its initial state, it will pick **the same** relabelings.

Thus, the verifier can recover the whole coloring simply by asking for the colors of the nodes one by one and resetting the prover between the iterations.

Can we fix this protocol?

# The fix

The idea behind the fix is as follows.

1. First, the verifier commits to the sequence of edges he is going to ask the prover to reveal.
2. The prover uses the commitments and a pseudo-random function to compute a random string.
3. The prover uses this random string to produce the random relabelings and commits to them.
4. The rest of the protocol proceeds in the natural way.

# Why does this work?

If the verifier resets the prover, he essentially has two choices.

- Commit to the same edge sequence. This produces the same transcript.
- Commit to a different edge sequence. The pseudo-random function ensures that the prover will use an independent sequence of colorings. This produces an independent transcript.

Thus, the verifier only gets a small benefit from resetting the prover. Essentially, he can run independent repetitions of the protocol.

The protocol can be shown to be resettable witness indistinguishable (**RWI**), but it is not (known to be) RZK.

# RZK more precisely

Let $P$ be a prover, $x_i$ and $y_i$ sequences of common inputs and auxiliary prover inputs, and $\omega_j$ be a sequence of random strings.

Let $P^{i,j} = P_{x_i,y_i,\omega_j}$ be the prover that result when we fix the common and auxiliary input, and the random string.

Each $P^{i,j}$ is called an **incarnation** or **clone** of $P$.

The verifier is allowed to interact with polynomially many different $P^{i,j}$s.

Each $P^{i,j}$ can be **instantiated** (e.g. reset) a polynomial number of times.

$P$ is RZK, if each probabilistic polynomial time verifier in this model can be simulated by a probabilistic polynomial time algorithm given only the $x_i$s.

# On the strength of RZK

RZK is the **strongest** notation of zero knowledge put forward to date.

It remains secure, even if an adversary is allowed to:

- Interact with **polynomially many independent** clones of the prover.
- **Interleave** the interaction with the clones in arbitrary ways.
- **Reset** each clone to its initial state polynomially many times.

Note that the clones are not aware of each other.

In particular, RZK implies **concurrent** zero knowledge.

# Sequential vs. interleaved

We can actually assume that the adversary interacts with the clones **sequentially**.

The adversary operates in rounds. At each round, he picks one of the clones and completes a full interaction with it.

Any adversary in the interleaved model can easily be simulated in the sequential model by rewinding the clone in a suitable way.

Thus, we can analyze adversaries in a much simpler sequential model.

# Constructing RZK protocols

Every language in $\mathbf{NP}$ has a constant round RWI protocol.

The construction is similar to the earlier example.

Every language in $\mathbf{NP}$ has a (non-constant round) RZK protocol.

Idea behind the construction: transform a witness indistinguishable protocol to a zero knowledge one.

# Constructing ZK from WI

The construction [Richardson, Killian 1999] has two stages.

- $P$ and $V$ runs a coin tossing into the well protocol. $P$ obtains $k$ uniformly distributed strings of length $k$, where $k$ is a security parameter.
- $P$ provides a WI proof that either some bit string is all zero, or $x \in L$.

If $k$ is chosen properly, the probability that some bit string is all zero is negligible. Thus, this is a proof system for $L$.

The purpose of the first stage is to give a **trapdoor** for the simulator.

The simulator can force one of the strings to be all zero. This gives the simulator a witness for the WI protocol.

# Constructing resettable protocols

First, the normal zero knowledge proof system for GRAPH 3-COLORING is transformed to a version where the verifier first commits to the edge sequence it is going to ask the prover to reveal. This step essentially determines all subsequent verifier moves.

Does **concurrency** help the verifier?

Zero knowledge proofs are also witness indistinguishable.

WI proofs are closed under concurrent composition.

The prover uses a pseudo random function to derive his randomness from the commitments.

The resulting RWI protocol is transformed to a RZK protocol using the previous transformation. All verifier commitments are moved to the first stage.

# The public key model

The public key model allows **constant round** RZK protocols for $\mathbf{NP}$.

In the public key model, the **verifiers** have public keys.

The only non-constant round part of the previous RZK protocol is Stage 1.

The only purpose of Stage 1 is to provide a trapdoor for the simulator.

This can be implemented in a constant number of rounds in the public key model.

The trapdoor is constructed using a **trapdoor commitment scheme**. The commitment receiver can open the commitment in an arbitrary way.

The **soundness** is more involved in the public key model [Micali, Reyzin 2001].