**University of Michigan–Ann Arbor**

Department of Electrical Engineering and Computer Science

EECS 475 **Introduction to Cryptography**, Winter 2023

**Lecture 25: Digital signatures based on discrete log: Identification Schemes, Schnorr's identification, Fiat-Shamir**

April 10, 2023

Lecturer: Mahdi Cheraghchi

Scribe: Yi-Wen Tseng

# 1 Identification Schemes

We can apply the hardness of discrete log to construct digital signatures, which is originally posted by Diffie Hellmen.
It is tricky and requires a detour into so-called **Identification Scheme**.

## 1.1 Goal

**Identification Scheme**: Prove that you have the key without revealing it.

## 1.2 Set-Up

We want to achieve that goal based on the hardness of discrete log.
We have a huge group $G$ of known order $q$. $G$ is a cyclic group, with order $|G| = q$, and generator $g$:

$$\mathbb{G} = \{g^0, g^1, g^2, \ldots, g^{q-1}\} =$$

Suppose $q$ is prime (for instance, think about $\mathbb{Z}_p^*$, $p = 2q + 1$ is prime.)
We want:

- Secrect key to be a random $x \in \mathbb{Z}_q$
- Public key would be $y = g^x$

Schnorr provided an interactive protocol between a prover (who has $x$) and a verifier (who only has $y$)
Prover wants to prove the knowledge of $x$ without revealing $x$. We need to show the correctness and the soundness of this model:

- **Correctness**:
  If $P, V$ run the protocol honestly, ($y = g^x$), then $V$ accepts.

$$g^s = g^{rx+k} = g^k \cdot (g^x)^r = c \cdot y^r$$

- **Soundness**:
  We need to make sure that if $V$ accepts (w.h.p), then $P$ knows $x$.
  Thought experiment: Consider two challenges $r_1 \neg r_2$ sent by $V$ to $P$, for which $P$ menages to make $V$ accept.
  Say $S_1, S_2$ are the responses by $P$.
  We know that if $g^{S_1} = c \cdot y^{r_1}$ and $g^{S_2} = c \cdot y^{r_2}$, then

$$g^{S_1 - S_2} = y^{r_1 - r_2} = g^{x(r_1 - r_2)}$$

$$s_1 - s_2 = x(r_1 - r_2) \bmod q$$

$$x = (s_1 - s_2)(r_1 - r_2)^{-1} \bmod q$$

  so we extract $x$ from the program runs $P$ against $V$.

## 2 Zero Knowledge

**Claim**: There is an efficient "simulator" that can efficiently sample from the distribution of the exchanged information without knowing $x$.
**Trick**: Sample from the joint distribution $(c, r, s)$ in this order: first $r$, then $s$, then $c$ in the way below:

1. Choose $r \leftarrow \mathbb{Z}_q$ uniformly
2. Choose $s \leftarrow \mathbb{Z}_q$ uniformly
3. Set $c = g^s \cdot y^{-r} \in G$

This has exactly the correct distribution but knows nothing about $x$ that $V$ doesn't. In other words, $V$ learns nothing new about $x$ other than the fact that $P$ knows it. There are two main issues in

this construction:

1. We want to sign stuff
2. We do not want interaction

**Fiat-Shamir Transform**: use a hash function
Signature Scheme: $(Gen, Sign, Ver)$

- Gen: Choose $x \leftarrow \mathbb{Z}_q$, output $(vk = y = g^x \in G, sk = x)$
- Sign($sk = x, m \in \{0,1\}^*$): Choose $k \leftarrow \mathbb{Z}_q$, let $c = g^k \in G$.
- Compute $r = H(m, c)$, where $H$ is really a "random oracle" and $s = k + r \cdot x \bmod q$. Output $\sigma = (r, s)$
- Ver($vk = y, m, \sigma = (r, s)$): compute $c = g^s \cdot y^{-r}$ and accept if $H(m, c) = r$.

**Theorem**: If discrete log is hard on $G$ and $H$ is a hash function modeled as a random oracle, then this is unforgeable.
**Proof Idea**: Because $H$ is a random oracle, the values $r = H(m, c)$ that a forger must deal with like truly random challenges in Schnorr's ID protocol. A forger will not be able to answer a challenge unless:

1. It gets extremely lucky in receiving the one challenge it knows how to handle
2. It is able to compute $x = \log_g(y)$ for the legit signer's public key $y$. (a uniform random element of $G$).