Schnorr Signature.

October 31, 2012

Table of contents

Schnorr Signature - Salient Features

- Derived from Schnorr identification scheme through Fiat-Shamir transformation
- Based on the DLP
- Security argued using oracle replay attacks
- Uses the random oracle heuristic

PRELIMINARIES

Proof through Contradiction

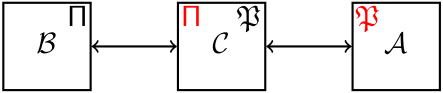
- Consider a protocol $\mathfrak{P}$ based on a *hard problem* Π

Proof through Contradiction

- Consider a protocol $\mathfrak{P}$ based on a *hard problem* Π
- Aim: Π is hard $\implies$ $\mathfrak{P}$ is not breakable

Proof through Contradiction

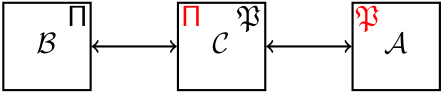
- Consider a protocol $\mathfrak{P}$ based on a *hard problem* Π
- Aim: Π is hard $\implies$ $\mathfrak{P}$ is not breakable $\equiv$
  $\mathfrak{P}$ is breakable $\implies$ Π is not hard



Proof through Contradiction

- Consider a protocol $\mathfrak{P}$ based on a *hard problem* Π
- Aim: Π is hard $\implies$ $\mathfrak{P}$ is not breakable $\equiv$

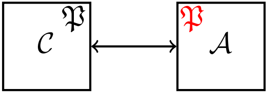
  $\mathfrak{P}$ is breakable $\implies$ Π is not hard



- Since Π is assumed to be hard, this leads to a *contradiction*.

Security Model

- Lays down the schema to be followed for giving security proofs
- Described using a game between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$



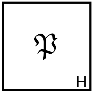
- $\mathcal{C}$ simulates the protocol environment for $\mathcal{A}$
- $\mathcal{A}$ wins the game if it solves the challenge given by $\mathcal{C}$

Random Oracles

- Heuristic aimed at simplifying security proofs of protocols involving hash functions.
- In proofs, the hash function modelled as a *truly random function* under the *control* of the challenger.
- $\mathcal{A}$ given oracle access to this function.

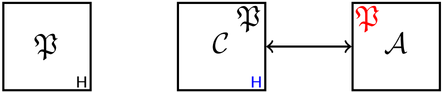
Random Oracles

- Heuristic aimed at simplifying security proofs of protocols involving hash functions.
- In proofs, the hash function modelled as a *truly random function* under the *control* of the challenger.
- $\mathcal{A}$ given oracle access to this function.



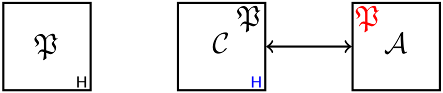
Random Oracles

- Heuristic aimed at simplifying security proofs of protocols involving hash functions.
- In proofs, the hash function modelled as a *truly random function* under the *control* of the challenger.
- $\mathcal{A}$ given oracle access to this function.



Random Oracles

- Heuristic aimed at simplifying security proofs of protocols involving hash functions.
- In proofs, the hash function modelled as a *truly random function* under the *control* of the challenger.
- $\mathcal{A}$ given oracle access to this function.



- Proofs without random oracles preferred.

PUBLIC-KEY SIGNATURES AND ITS SECURITY MODELS

Definition – Public-Key Signature

An PKS scheme consists of three PPT algorithms $\{\mathcal{K}, \mathcal{S}, \mathcal{V}\}$ -

Definition – Public-Key Signature

An PKS scheme consists of three PPT algorithms $\{\mathcal{K}, \mathcal{S}, \mathcal{V}\}$ -

- Key Generation:
  - Used by the user to generate the public-private key pair $(\mathsf{pk}, \mathsf{sk})$
  - pk is published and the sk kept secret
  - Run on a security parameter κ

$$(\mathsf{pk}, \mathsf{sk}) \xleftarrow{\$} \mathcal{K}(\kappa)$$

Definition – Public-Key Signature

An PKS scheme consists of three PPT algorithms $\{\mathcal{K}, \mathcal{S}, \mathcal{V}\}$ -

- Key Generation:
  - Used by the user to generate the public-private key pair $(\mathsf{pk}, \mathsf{sk})$
  - pk is published and the sk kept secret
  - Run on a security parameter κ

$$(\mathsf{pk}, \mathsf{sk}) \xleftarrow{\$} \mathcal{K}(\kappa)$$

- Signing:
  - Used by the user to generate signature on some message m
  - The secret key sk used for signing

$$\sigma \xleftarrow{\$} \mathcal{S}(\mathsf{sk}, m)$$

Definition – Public-Key Signature

An PKS scheme consists of three PPT algorithms $\{\mathcal{K}, \mathcal{S}, \mathcal{V}\}$ -

- Key Generation:
  - Used by the user to generate the public-private key pair $(\mathsf{pk}, \mathsf{sk})$
  - pk is published and the sk kept secret
  - Run on a security parameter κ

$$(\mathsf{pk}, \mathsf{sk}) \xleftarrow{\$} \mathcal{K}(\kappa)$$

- Signing:
  - Used by the user to generate signature on some message m
  - The secret key sk used for signing

$$\sigma \xleftarrow{\$} \mathcal{S}(\mathsf{sk}, m)$$

- Verification:
  - Outputs 1 if σ is a valid signature on m; else, outputs 0

$$\mathsf{result} \leftarrow \mathcal{V}(\sigma, m, \mathsf{pk})$$

Definition – EU-NMA

- *Existential unforgeability under no-message attack*

Definition – EU-NMA

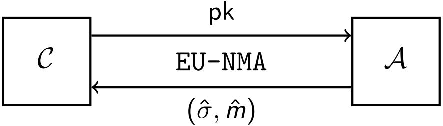
- *Existential unforgeability under no-message attack*
- Challenger $\mathcal{C}$ generates key-pair $(\mathsf{pk}, \mathsf{sk})$.

Definition – EU-NMA

- *Existential unforgeability under no-message attack*
- Challenger $\mathcal{C}$ generates key-pair $(\mathsf{pk}, \mathsf{sk})$.

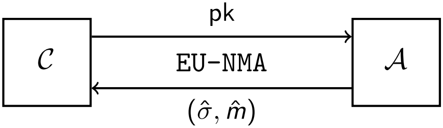
Definition – EU-NMA

- *Existential unforgeability under no-message attack*
- Challenger $\mathcal{C}$ generates key-pair $(\mathsf{pk}, \mathsf{sk})$.
- Forgery – Adversary $\mathcal{A}$ wins if $\hat{\sigma}$ is a *valid* signature on $\hat{m}$.



Definition – EU-NMA

- *Existential unforgeability under no-message attack*
- Challenger $\mathcal{C}$ generates key-pair $(\mathsf{pk}, \mathsf{sk})$.
- Forgery – Adversary $\mathcal{A}$ wins if $\hat{\sigma}$ is a *valid* signature on $\hat{m}$.

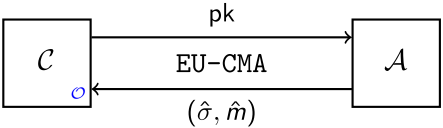


- Adversary's advantage in the game:

$$\Pr\left[1 \leftarrow \mathcal{V}(\hat{\sigma}, \hat{m}, \mathsf{pk}) \mid (\mathsf{sk}, \mathsf{pk}) \xleftarrow{\$} \mathcal{K}(\kappa); (\hat{\sigma}, \hat{m}) \xleftarrow{\$} \mathcal{A}(\mathsf{pk})\right]$$

Definition – EU-CMA

- *Existential unforgeability under chosen-message attack*
- Challenger $\mathcal{C}$ generates key-pair $(\mathsf{pk}, \mathsf{sk})$.
- Signature Queries – Access to a signing oracle $\mathcal{O}$
- Forgery – Adversary $\mathcal{A}$ wins if
  - $\hat{\sigma}$ is a *valid* signature on $\hat{m}$.
  - $\mathcal{A}$ has *not* made a signature query on $\hat{m}$.

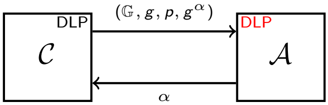


- Adversary's advantage in the game:

$$\Pr\left[1 \leftarrow \mathcal{V}(\hat{\sigma}, \hat{m}, \mathsf{pk}) \mid (\mathsf{sk}, \mathsf{pk}) \xleftarrow{\$} \mathcal{K}(\kappa); (\hat{\sigma}, \hat{m}) \xleftarrow{\$} \mathcal{A}^{\mathcal{O}}(\mathsf{pk})\right]$$

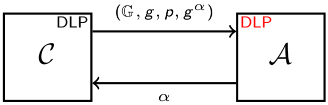
Hardness Assumption: Discrete-log Assumption

Discrete-log problem for a group $\mathbb{G} = \langle g \rangle$ and $|\mathbb{G}| = p$



Hardness Assumption: Discrete-log Assumption

Discrete-log problem for a group $\mathbb{G} = \langle g \rangle$ and $|\mathbb{G}| = p$



Definition

The DLP in $\mathbb{G}$ is to find α given g^{α}, where $\alpha \in_R \mathbb{Z}_p$. An adversary $\mathcal{A}$ has advantage ϵ in solving the DLP if

$$\Pr\left[\alpha \in_R \mathbb{Z}_p; \alpha' \leftarrow \mathcal{A}(\mathbb{G}, p, g, g^{\alpha}) \mid \alpha' = \alpha\right] \geq \epsilon.$$

The (ϵ, t)-discrete-log assumption holds in $\mathbb{G}$ if no adversary has advantage at least ϵ in solving the DLP in time at most t.

SCHNORR SIGNATURE

Schnorr Signature

The Setting.

1. We work in group $\mathbb{G} = \langle g \rangle$ of prime order p.
2. A hash function H is used.

$$\mathsf{H} : \{0,1\}^* \to \mathbb{Z}_p$$

Schnorr Signature

The Setting.

1. We work in group $\mathbb{G} = \langle g \rangle$ of prime order p.
2. A hash function H is used.

$$\mathsf{H} : \{0,1\}^* \to \mathbb{Z}_p$$

Key Generation. $\mathcal{K}(\kappa)$:

1. Select $z \in_R \mathbb{Z}_p$ as the secret key sk
2. Set $Z := g^z$ as the public key pk

Schnorr Signature

The Setting.

1. We work in group $\mathbb{G} = \langle g \rangle$ of prime order p.
2. A hash function H is used.

$$\mathsf{H} : \{0,1\}^* \to \mathbb{Z}_p$$

Key Generation. $\mathcal{K}(\kappa)$:

1. Select $z \in_R \mathbb{Z}_p$ as the secret key sk
2. Set $Z := g^z$ as the public key pk

Signing. $\mathcal{S}(m, \mathsf{sk})$:

1. Let $\mathsf{sk} = z$. Select $r \in_R \mathbb{Z}_p$, set $R := g^r$ and $c := \mathsf{H}(m, R)$.
2. The signature on m is $\sigma := (y, R)$ where

$$y := r + zc$$

Schnorr Signature

Verification. $\mathcal{V}(\sigma, m)$:

1. Let $\sigma = (y, R)$ and $c = \mathsf{H}(m, R)$.
2. σ is valid if

$$g^y = RZ^c$$

Security of Schnorr Signature: An Intuition

- Consider an adversary $\mathcal{A}$ with ability to launch chosen-message attack on the Schnorr signature.
- Let $\{\sigma_1, \ldots, \sigma_n\}$ with $\sigma_i = (r_i + zc_i, R_i)$ on m_i be the signatures that $\mathcal{A}$ receives.

Security of Schnorr Signature: An Intuition

- Consider an adversary $\mathcal{A}$ with ability to launch chosen-message attack on the Schnorr signature.
- Let $\{\sigma_1, \ldots, \sigma_n\}$ with $\sigma_i = (r_i + zc_i, R_i)$ on m_i be the signatures that $\mathcal{A}$ receives.

$$\begin{pmatrix} 1 & 0 & \cdots & 0 & c_0 \\ 0 & 1 & \cdots & 0 & c_1 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & c_n \end{pmatrix} \times \begin{pmatrix} r_1 \\ r_2 \\ \vdots \\ r_n \\ z \end{pmatrix} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ r_n \end{pmatrix}$$

Security of Schnorr Signature: An Intuition

- *However*, $\mathcal{A}$ can solve for x if it gets two equations containing the *same* r but *different* c, i.e.

$$y_1 = r + zc_1 \quad \text{and} \quad y_2 = r + zc_2$$

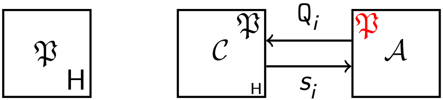
implies

$$z = \frac{y_1 - y_2}{c_1 - c_2}$$

ORACLE REPLAY ATTACK

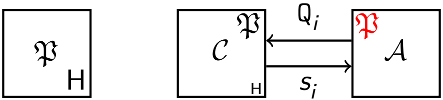
The Oracle Replay Attack

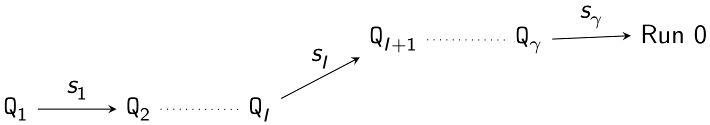
- Recall the random oracle methodology.



The Oracle Replay Attack

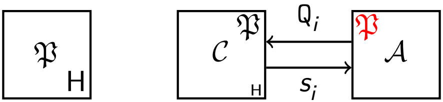
- Recall the random oracle methodology.

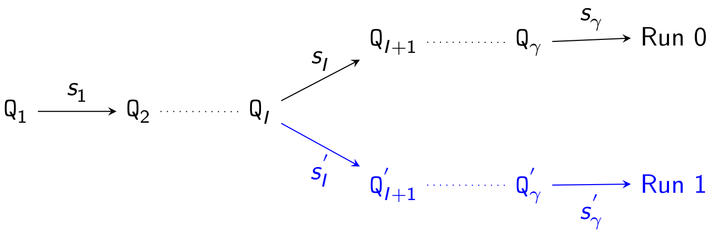




The Oracle Replay Attack

- Recall the random oracle methodology.

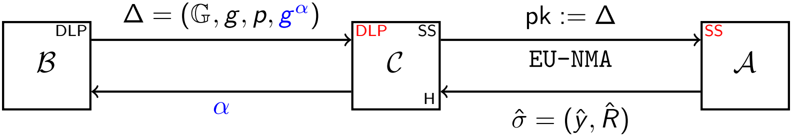




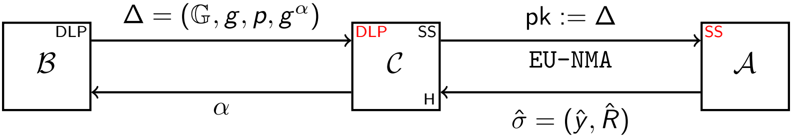
- The simulation carried out during Run 1 (from query $\mathtt{Q}_i$) using a *different* random function

Security of Schnorr Signature in EU-NMA

- Consider the simpler model of existential unforgeability under no-message attack (EU-NMA)
  - $\mathcal{C}$ gives the challenge public key $\mathsf{pk} := (\mathbb{G}, g, p, g^{\alpha})$ to $\mathcal{A}$
  - $\mathcal{A}$ not allowed signature queries; forges on a message $\hat{m}$
  - $\mathcal{A}$ also allowed access to an H-oracle $\{\mathtt{Q}_1, \ldots, \mathtt{Q}_\gamma\}$



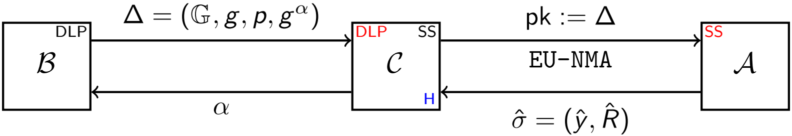
Security of Schnorr Signature in EU-NMA



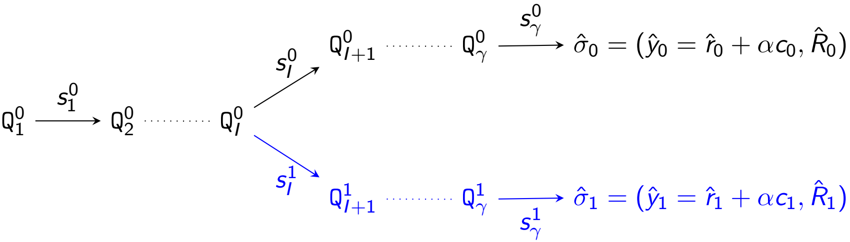
- $\mathsf{Q}_I^0 : \mathsf{H}(\hat{m}_0, \hat{R}_0) = c_0$

$$\mathsf{Q}_1^0 \xrightarrow{s_1^0} \mathsf{Q}_2^0 \cdots\cdots \mathsf{Q}_I^0 \xrightarrow{s_I^0} \mathsf{Q}_{I+1}^0 \cdots\cdots \mathsf{Q}_\gamma^0 \xrightarrow{s_\gamma^0} \hat{\sigma}_0 = (\hat{y}_0 = \hat{r}_0 + \alpha c_0, \hat{R}_0)$$

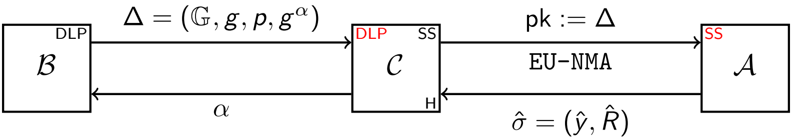
Security of Schnorr Signature in EU-NMA



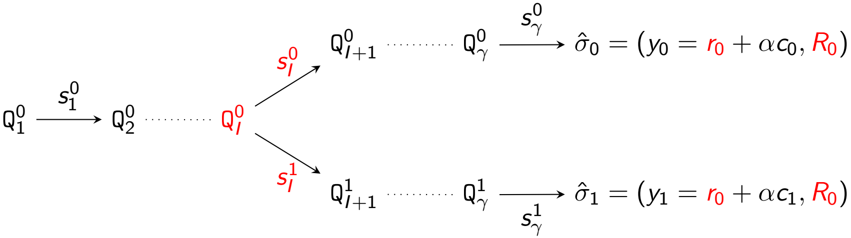
- $\mathtt{Q}_I^0 : \mathsf{H}(\hat{m}_0, \hat{R}_0) = c_0$ and $\mathtt{Q}_J^{\phi} : \mathsf{H}(\hat{m}_1, \hat{R}_1) = c_1$



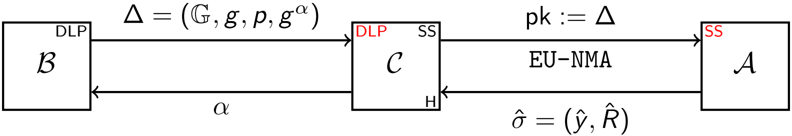
Security of Schnorr Signature in EU-NMA



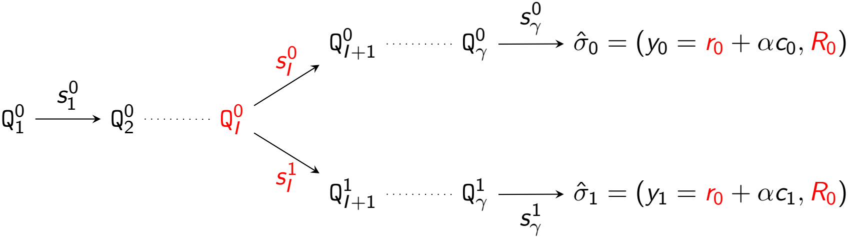
- $J = I \implies \hat{m}_1 = \hat{m}_0 \wedge \hat{R}_1 = \hat{R}_0$



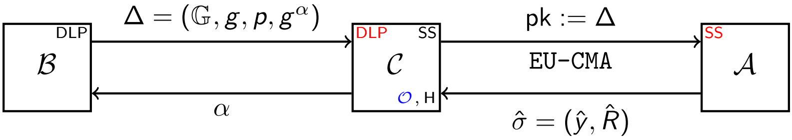
Security of Schnorr Signature in EU-NMA



- $J = I \implies \hat{m}_1 = \hat{m}_0 \wedge \hat{R}_1 = \hat{R}_0$ and $\alpha = (\hat{y}_0 - \hat{y}_1)/(c_0 - c_1)$

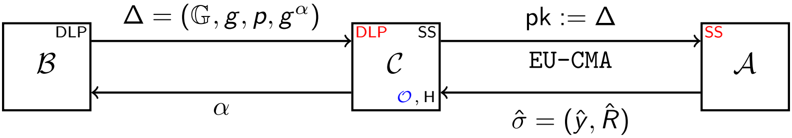


Security in the Full Model



Signature query. $\mathcal{O}(m)$

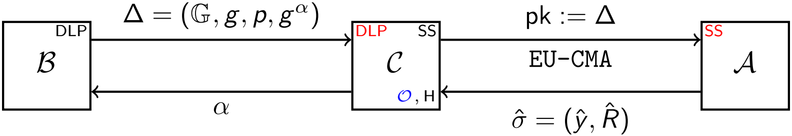
Security in the Full Model



Signature query. $\mathcal{O}(m)$

- Signature for m is of form $\sigma = (r + \alpha c, g^r)$, where $c = \mathsf{H}(m, g^r)$.

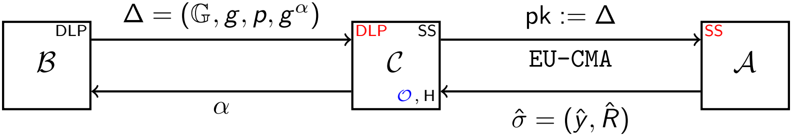
Security in the Full Model



Signature query. $\mathcal{O}(m)$

- Signature for m is of form $\sigma = (r + \alpha c, g^r)$, where $c = \mathsf{H}(m, g^r)$.
- But, we don't know α!

Security in the Full Model



Signature query. $\mathcal{O}(m)$

- Signature for m is of form $\sigma = (r + \alpha c, g^r)$, where $c = \mathsf{H}(m, g^r)$.
- But, we don't know α!
- Problem solved using Boneh-Boyen algebraic technique:
  - Select $c, s \in_R \mathbb{Z}_p$ and set $r = -\alpha c + s$.
  - Program the random oracle to set $c = \mathsf{H}(m, g^r)$ and send (s, g^r) as the signature.

FORKING LEMMA

Forking Algorithm

- The oracle replay attack formalised through the *forking algorithm*

Algorithm 1 $\mathcal{F}_{\mathcal{Y}}(x)$

Pick coins ρ for $\mathcal{Y}$ at random
$s_1^0, \ldots, s_\gamma^0 \in_R \mathbb{S}$; $(I_0, \sigma_0) \xleftarrow{\$} \mathcal{Y}(x, s_1^0, \ldots, s_\gamma^0; \rho)$ [Run 0]
$s_{I_0}^1, \ldots, s_\gamma^1 \in_R \mathbb{S}$; $(I_1, \sigma_1) \xleftarrow{\$} \mathcal{Y}(x, s_1^0, \ldots, s_{I_0-1}^0, s_{I_0}^1, \ldots, s_\gamma^1; \rho)$ [Run 1]
if $(I_0 > 0 \wedge I_1 = I_0 \wedge s_{I_0}^1 \neq s_{I_0}^0)$ **then**
  return $(1, \sigma_0, \sigma_1)$
else
  return $(0, \perp, \perp)$
end if

The Forking Lemma

- The *forking lemma* gives a lower bound on the success probability of the oracle replay attack (frk) in terms of the success probability of the adversary during a particular run (acc).

The Forking Lemma

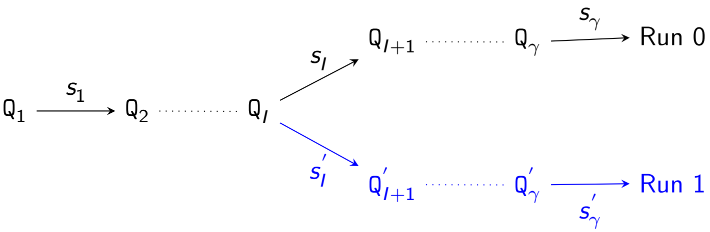
- The *forking lemma* gives a lower bound on the success probability of the oracle replay attack (frk) in terms of the success probability of the adversary during a particular run (acc).
- To be precise,

$$\mathrm{frk} \geq \mathrm{acc}\left(\frac{\mathrm{acc}}{\gamma} - \frac{1}{p}\right)$$

The Forking Lemma

- The *forking lemma* gives a lower bound on the success probability of the oracle replay attack (frk) in terms of the success probability of the adversary during a particular run (acc).
- To be precise,

$$\mathrm{frk} \geq \mathrm{acc}\left(\frac{\mathrm{acc}}{\gamma} - \frac{1}{p}\right)$$



Theorem

Theorem

If $\mathcal{A}$ is an adversary with advantage ϵ against the Schnorr signature scheme, in the setting $(\mathbb{G}, g, p)$, then we can construct an algorithm $\mathcal{B}$ that solves the DLP with advantage

$$\epsilon' \geq \epsilon \left(\frac{\epsilon}{\gamma} - \frac{1}{p} \right)$$

provided H is modelled as a random oracle with an upper bound of q_H queries.

Related Literature

1. Mihir Bellare and Gregory Neven. Multi-signatures in the plain public-key model and a general forking lemma.
2. David Pointcheval and Jacques Stern. Security arguments for digital signatures and blind signatures.
3. Sanjam Garg, Raghav Bhaskar, and Satyanarayana V. Lokam. Improved bounds on security reductions for discrete log based signatures.
4. Yannick Seurin. On the exact security of Schnorr-type signatures in the random oracle model.

QUESTIONS?

THANK YOU!