On the Exact Security of Schnorr-Type Signatures in the Random Oracle Model

Yannick Seurin

ANSSI, France

18 April, EUROCRYPT 2012

Introduction

- Schnorr signatures: best-known example of the Fiat-Shamir heuristic
- proven secure (under the DL assumption) in the Random Oracle Model by Pointcheval and Stern (EC '96) with the Forking Lemma
- security reduction loses a factor q_h (number of RO queries of the forger), potentially very large
- previous results showed that losing some factor was "unavoidable":
  - a $q_h^{1/2}$ factor (Paillier and Vergnaud, AC 2005)
  - a $q_h^{2/3}$ factor (Garg, Bhaskar, and Lokam, CRYPTO 2008)
- we show that losing a q_h factor is unavoidable, closing the gap between the Forking Lemma and previous impossibility results

Introduction

- Schnorr signatures: best-known example of the Fiat-Shamir heuristic
- proven secure (under the DL assumption) in the Random Oracle Model by Pointcheval and Stern (EC '96) with the Forking Lemma
- security reduction loses a factor q_h (number of RO queries of the forger), potentially very large
- previous results showed that losing some factor was "unavoidable":
  - a $q_h^{1/2}$ factor (Paillier and Vergnaud, AC 2005)
  - a $q_h^{2/3}$ factor (Garg, Bhaskar, and Lokam, CRYPTO 2008)
- we show that losing a q_h factor is unavoidable, closing the gap between the Forking Lemma and previous impossibility results

Introduction

- Schnorr signatures: best-known example of the Fiat-Shamir heuristic
- proven secure (under the DL assumption) in the Random Oracle Model by Pointcheval and Stern (EC '96) with the Forking Lemma
- security reduction loses a factor q_h (number of RO queries of the forger), potentially very large
- previous results showed that losing some factor was "unavoidable":
  - a $q_h^{1/2}$ factor (Paillier and Vergnaud, AC 2005)
  - a $q_h^{2/3}$ factor (Garg, Bhaskar, and Lokam, CRYPTO 2008)
- we show that losing a q_h factor is unavoidable, closing the gap between the Forking Lemma and previous impossibility results

Introduction

- Schnorr signatures: best-known example of the Fiat-Shamir heuristic
- proven secure (under the DL assumption) in the Random Oracle Model by Pointcheval and Stern (EC '96) with the Forking Lemma
- security reduction loses a factor q_h (number of RO queries of the forger), potentially very large
- previous results showed that losing some factor was "unavoidable":
  - a $q_h^{1/2}$ factor (Paillier and Vergnaud, AC 2005)
  - a $q_h^{2/3}$ factor (Garg, Bhaskar, and Lokam, CRYPTO 2008)
- we show that losing a q_h factor is unavoidable, closing the gap between the Forking Lemma and previous impossibility results

Introduction

- Schnorr signatures: best-known example of the Fiat-Shamir heuristic
- proven secure (under the DL assumption) in the Random Oracle Model by Pointcheval and Stern (EC '96) with the Forking Lemma
- security reduction loses a factor q_h (number of RO queries of the forger), potentially very large
- previous results showed that losing some factor was "unavoidable":
  - a $q_h^{1/2}$ factor (Paillier and Vergnaud, AC 2005)
  - a $q_h^{2/3}$ factor (Garg, Bhaskar, and Lokam, CRYPTO 2008)
- we show that losing a q_h factor is unavoidable, closing the gap between the Forking Lemma and previous impossibility results

Introduction

- Schnorr signatures: best-known example of the Fiat-Shamir heuristic
- proven secure (under the DL assumption) in the Random Oracle Model by Pointcheval and Stern (EC '96) with the Forking Lemma
- security reduction loses a factor q_h (number of RO queries of the forger), potentially very large
- previous results showed that losing some factor was "unavoidable":
  - a $q_h^{1/2}$ factor (Paillier and Vergnaud, AC 2005)
  - a $q_h^{2/3}$ factor (Garg, Bhaskar, and Lokam, CRYPTO 2008)
- we show that losing a q_h factor is unavoidable, closing the gap between the Forking Lemma and previous impossibility results

Introduction

- Schnorr signatures: best-known example of the Fiat-Shamir heuristic
- proven secure (under the DL assumption) in the Random Oracle Model by Pointcheval and Stern (EC '96) with the Forking Lemma
- security reduction loses a factor q_h (number of RO queries of the forger), potentially very large
- previous results showed that losing some factor was "unavoidable":
  - a $q_h^{1/2}$ factor (Paillier and Vergnaud, AC 2005)
  - a $q_h^{2/3}$ factor (Garg, Bhaskar, and Lokam, CRYPTO 2008)
- we show that losing a q_h factor is unavoidable, closing the gap between the Forking Lemma and previous impossibility results

Outline

1. Schnorr Signatures and The Forking Lemma
2. Meta-Reductions
3. Main Result

Outline

1. Schnorr Signatures and The Forking Lemma
2. Meta-Reductions
3. Main Result

Schnorr signatures

- $\mathbb{G}$ cyclic group of prime order q and G a generator of $\mathbb{G}$
- secret key: $x \in_{\mathrm{r}} \mathbb{Z}_q \setminus \{0\}$
- public key: $X = G^x$
- $\mathtt{Sign}(m)$, $m \in \{0,1\}^*$:
  - $a \in_{\mathrm{r}} \mathbb{Z}_q$, $A = G^a$ (commitment) $\xrightarrow{A=G^a}$
  - $c = H(m, A)$ (challenge) $\xleftarrow{c}$
  - $s = a + cx \mod q$ (answer) $\xrightarrow{s=a+cx}$
  - signature is (s, c)
- $\mathtt{Verif}(m, (s, c))$:
  - $A = G^s X^{-c}$
  - check $H(m, A) = c$

Here H is modeled as a random oracle $\boldsymbol{H}$

Schnorr signatures

- $\mathbb{G}$ cyclic group of prime order q and G a generator of $\mathbb{G}$
- secret key: $x \in_{\mathrm{r}} \mathbb{Z}_q \setminus \{0\}$
- public key: $X = G^x$
- $\mathtt{Sign}(m)$, $m \in \{0,1\}^*$:
  - $a \in_{\mathrm{r}} \mathbb{Z}_q$, $A = G^a$ (commitment) $\xrightarrow{A=G^a}$
  - $c = H(m, A)$ (challenge) $\xleftarrow{c}$
  - $s = a + cx \mod q$ (answer) $\xrightarrow{s=a+cx}$
  - signature is (s, c)
- $\mathtt{Verif}(m, (s, c))$:
  - $A = G^s X^{-c}$
  - check $H(m, A) = c$

Here H is modeled as a random oracle $\boldsymbol{H}$

Schnorr signatures

- $\mathbb{G}$ cyclic group of prime order q and G a generator of $\mathbb{G}$
- secret key: $x \in_{\mathrm{r}} \mathbb{Z}_q \setminus \{0\}$
- public key: $X = G^x$
- $\mathtt{Sign}(m)$, $m \in \{0,1\}^*$:
  - $a \in_{\mathrm{r}} \mathbb{Z}_q$, $A = G^a$ (commitment) $\xrightarrow{A=G^a}$
  - $c = H(m, A)$ (challenge) $\xleftarrow{c}$
  - $s = a + cx \mod q$ (answer) $\xrightarrow{s=a+cx}$
  - signature is (s, c)
- $\mathtt{Verif}(m, (s, c))$:
  - $A = G^s X^{-c}$
  - check $H(m, A) = c$

Here H is modeled as a random oracle $\boldsymbol{H}$

Schnorr signatures

- $\mathbb{G}$ cyclic group of prime order q and G a generator of $\mathbb{G}$
- secret key: $x \in_{\mathrm{r}} \mathbb{Z}_q \setminus \{0\}$
- public key: $X = G^x$
- $\mathtt{Sign}(m)$, $m \in \{0,1\}^*$:
  - $a \in_{\mathrm{r}} \mathbb{Z}_q$, $A = G^a$ (commitment) $\xrightarrow{A=G^a}$
  - $c = H(m, A)$ (challenge) $\xleftarrow{c}$
  - $s = a + cx \mod q$ (answer) $\xrightarrow{s=a+cx}$
  - signature is (s, c)
- $\mathtt{Verif}(m, (s, c))$:
  - $A = G^s X^{-c}$
  - check $H(m, A) = c$

Here H is modeled as a random oracle $\boldsymbol{H}$

Schnorr signatures

- $\mathbb{G}$ cyclic group of prime order q and G a generator of $\mathbb{G}$
- secret key: $x \in_{\mathrm{r}} \mathbb{Z}_q \setminus \{0\}$
- public key: $X = G^x$
- $\mathtt{Sign}(m)$, $m \in \{0,1\}^*$:
  - $a \in_{\mathrm{r}} \mathbb{Z}_q$, $A = G^a$ (commitment) $\xrightarrow{A=G^a}$
  - $c = H(m, A)$ (challenge) $\xleftarrow{c}$
  - $s = a + cx \mod q$ (answer) $\xrightarrow{s=a+cx}$
  - signature is (s, c)
- $\mathtt{Verif}(m, (s, c))$:
  - $A = G^s X^{-c}$
  - check $H(m, A) = c$

Here H is modeled as a random oracle $\boldsymbol{H}$

Schnorr signatures

- $\mathbb{G}$ cyclic group of prime order q and G a generator of $\mathbb{G}$
- secret key: $x \in_r \mathbb{Z}_q \setminus \{0\}$
- public key: $X = G^x$
- $\texttt{Sign}(m)$, $m \in \{0,1\}^*$:
  - $a \in_r \mathbb{Z}_q$, $A = G^a$ (commitment) $\xrightarrow{A=G^a}$
  - $c = H(m, A)$ (challenge) $\xleftarrow{c}$
  - $s = a + cx \mod q$ (answer) $\xrightarrow{s=a+cx}$
  - signature is (s, c)
- $\texttt{Verif}(m, (s, c))$:
  - $A = G^s X^{-c}$
  - check $H(m, A) = c$

Here H is modeled as a random oracle $\boldsymbol{H}$

Schnorr signatures

- $\mathbb{G}$ cyclic group of prime order q and G a generator of $\mathbb{G}$
- secret key: $x \in_{\mathrm{r}} \mathbb{Z}_q \setminus \{0\}$
- public key: $X = G^x$
- $\mathtt{Sign}(m)$, $m \in \{0,1\}^*$:
  - $a \in_{\mathrm{r}} \mathbb{Z}_q$, $A = G^a$ (commitment) $\xrightarrow{A=G^a}$
  - $c = H(m, A)$ (challenge) $\xleftarrow{c}$
  - $s = a + cx \mod q$ (answer) $\xrightarrow{s=a+cx}$
  - signature is (s, c)
- $\mathtt{Verif}(m, (s, c))$:
  - $A = G^s X^{-c}$
  - check $H(m, A) = c$

Here H is modeled as a random oracle $\boldsymbol{H}$

Schnorr signatures

- $\mathbb{G}$ cyclic group of prime order q and G a generator of $\mathbb{G}$
- secret key: $x \in_{\mathrm{r}} \mathbb{Z}_q \setminus \{0\}$
- public key: $X = G^x$
- $\mathtt{Sign}(m)$, $m \in \{0,1\}^*$:
  - $a \in_{\mathrm{r}} \mathbb{Z}_q$, $A = G^a$ (commitment) $\xrightarrow{A=G^a}$
  - $c = H(m, A)$ (challenge) $\xleftarrow{c}$
  - $s = a + cx \mod q$ (answer) $\xrightarrow{s=a+cx}$
  - signature is (s, c)
- $\mathtt{Verif}(m, (s, c))$:
  - $A = G^s X^{-c}$
  - check $H(m, A) = c$

Here H is modeled as a random oracle $\boldsymbol{H}$

Schnorr signatures

- $\mathbb{G}$ cyclic group of prime order q and G a generator of $\mathbb{G}$
- secret key: $x \in_{\mathrm{r}} \mathbb{Z}_q \setminus \{0\}$
- public key: $X = G^x$
- $\mathtt{Sign}(m)$, $m \in \{0,1\}^*$:
  - $a \in_{\mathrm{r}} \mathbb{Z}_q$, $A = G^a$ (commitment) $\xrightarrow{A=G^a}$
  - $c = H(m, A)$ (challenge) $\xleftarrow{c}$
  - $s = a + cx \mod q$ (answer) $\xrightarrow{s=a+cx}$
  - signature is (s, c)
- $\mathtt{Verif}(m, (s, c))$:
  - $A = G^s X^{-c}$
  - check $H(m, A) = c$

Here H is modeled as a random oracle $\boldsymbol{H}$

Schnorr signatures

- $\mathbb{G}$ cyclic group of prime order q and G a generator of $\mathbb{G}$
- secret key: $x \in_{\mathrm{r}} \mathbb{Z}_q \setminus \{0\}$
- public key: $X = G^x$
- $\mathtt{Sign}(m)$, $m \in \{0,1\}^*$:
  - $a \in_{\mathrm{r}} \mathbb{Z}_q$, $A = G^a$ (commitment) $\xrightarrow{A=G^a}$
  - $c = H(m, A)$ (challenge) $\xleftarrow{c}$
  - $s = a + cx \mod q$ (answer) $\xrightarrow{s=a+cx}$
  - signature is (s, c)
- $\mathtt{Verif}(m, (s, c))$:
  - $A = G^s X^{-c}$
  - check $H(m, A) = c$

Here H is modeled as a random oracle $\boldsymbol{H}$

Schnorr signatures

- $\mathbb{G}$ cyclic group of prime order q and G a generator of $\mathbb{G}$
- secret key: $x \in_{\mathrm{r}} \mathbb{Z}_q \setminus \{0\}$
- public key: $X = G^x$
- $\mathtt{Sign}(m)$, $m \in \{0,1\}^*$:
  - $a \in_{\mathrm{r}} \mathbb{Z}_q$, $A = G^a$ (commitment) $\xrightarrow{A=G^a}$
  - $c = H(m, A)$ (challenge) $\xleftarrow{c}$
  - $s = a + cx \mod q$ (answer) $\xrightarrow{s=a+cx}$
  - signature is (s, c)
- $\mathtt{Verif}(m, (s, c))$:
  - $A = G^s X^{-c}$
  - check $H(m, A) = c$

Here H is modeled as a random oracle $\boldsymbol{H}$

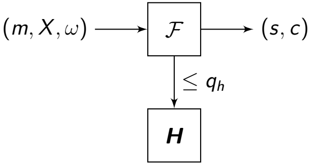
Schnorr signatures

- $\mathbb{G}$ cyclic group of prime order q and G a generator of $\mathbb{G}$
- secret key: $x \in_r \mathbb{Z}_q \setminus \{0\}$
- public key: $X = G^x$
- $\texttt{Sign}(m)$, $m \in \{0,1\}^*$:
  - $a \in_r \mathbb{Z}_q$, $A = G^a$ (commitment) $\xrightarrow{A=G^a}$
  - $c = H(m, A)$ (challenge) $\xleftarrow{c}$
  - $s = a + cx \mod q$ (answer) $\xrightarrow{s=a+cx}$
  - signature is (s, c)
- $\texttt{Verif}(m, (s, c))$:
  - $A = G^s X^{-c}$
  - check $H(m, A) = c$

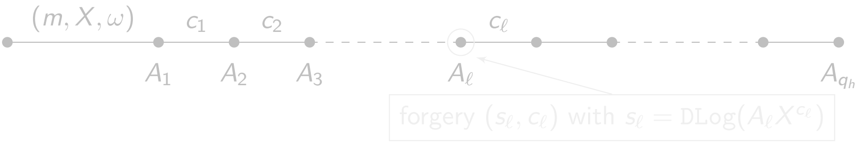
Here H is modeled as a random oracle $\boldsymbol{H}$

Forger adversary against Schnorr signatures

- we focus on universal forgery under no-message attacks: the adversary is given a message m and a public key X and must return a forgery (s, c) for m (it cannot make signature queries)
- the random tape of the forger will be explicitly denoted ω
- parameters characterizing a forger $\mathcal{F}$:
  - running time t_F
  - success probability ε_F
    $\rightarrow$ time-to-success ratio $\rho_F = t_F/\varepsilon_F$
  - maximal number of RO queries q_h

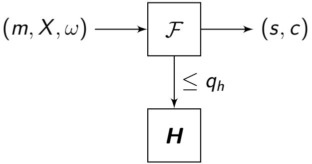


- pictorial representation of a forgery experiment:

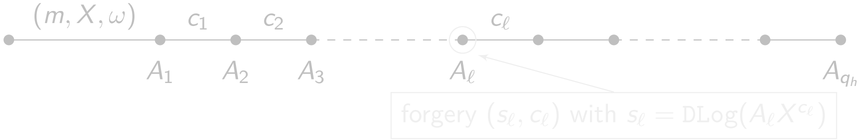


Forger adversary against Schnorr signatures

- we focus on universal forgery under no-message attacks: the adversary is given a message m and a public key X and must return a forgery (s, c) for m (it cannot make signature queries)
- the random tape of the forger will be explicitly denoted ω

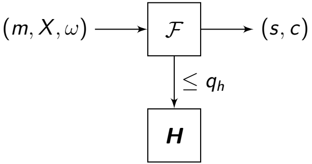


- parameters characterizing a forger $\mathcal{F}$:
  - running time t_F
  - success probability ε_F
    $\rightarrow$ time-to-success ratio $\rho_F = t_F / \varepsilon_F$
  - maximal number of RO queries q_h
- pictorial representation of a forgery experiment:

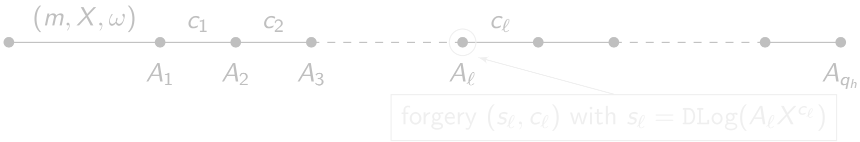


Forger adversary against Schnorr signatures

- we focus on universal forgery under no-message attacks: the adversary is given a message m and a public key X and must return a forgery (s, c) for m (it cannot make signature queries)
- the random tape of the forger will be explicitly denoted ω
- parameters characterizing a forger $\mathcal{F}$:
  - running time t_F
  - success probability ε_F
    $\rightarrow$ time-to-success ratio $\rho_F = t_F / \varepsilon_F$
  - maximal number of RO queries q_h

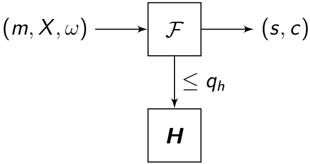


- pictorial representation of a forgery experiment:

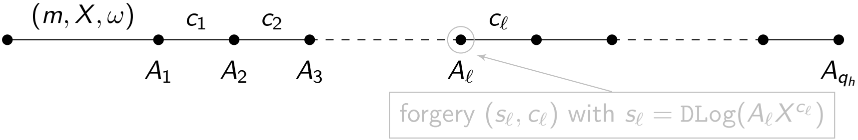


Forger adversary against Schnorr signatures

- we focus on universal forgery under no-message attacks: the adversary is given a message m and a public key X and must return a forgery (s, c) for m (it cannot make signature queries)
- the random tape of the forger will be explicitly denoted ω
- parameters characterizing a forger $\mathcal{F}$:
  - running time t_F
  - success probability ε_F
    $\rightarrow$ time-to-success ratio $\rho_F = t_F / \varepsilon_F$
  - maximal number of RO queries q_h



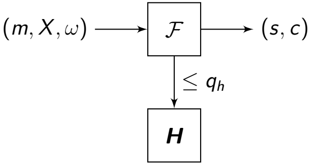
- pictorial representation of a forgery experiment:



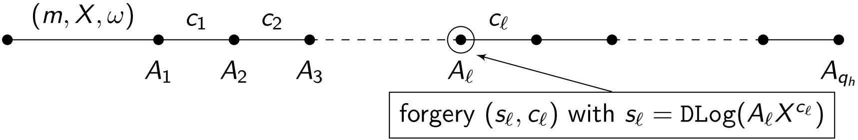
Forger adversary against Schnorr signatures

- we focus on universal forgery under no-message attacks: the adversary is given a message m and a public key X and must return a forgery (s, c) for m (it cannot make signature queries)
- the random tape of the forger will be explicitly denoted ω

- parameters characterizing a forger $\mathcal{F}$:
  - running time t_F
  - success probability ε_F
    $\rightarrow$ time-to-success ratio $\rho_F = t_F / \varepsilon_F$
  - maximal number of RO queries q_h



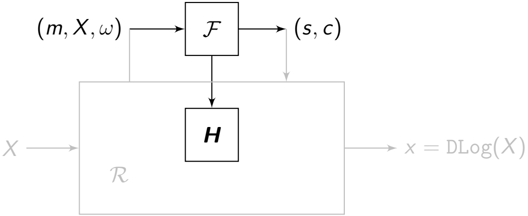
- pictorial representation of a forgery experiment:



Extracting discrete logarithms from a forger

- given a forger $\mathcal{F}$, one can build a reduction $\mathcal{R}$ which solves the DL problem for the public key $X = G^x$ using $\mathcal{F}$ as a black-box
- main idea: have the forger output two forgeries (s_1, c_1) and (s_2, c_2) for the same message m and the same commitment $A = G^a$, so that:

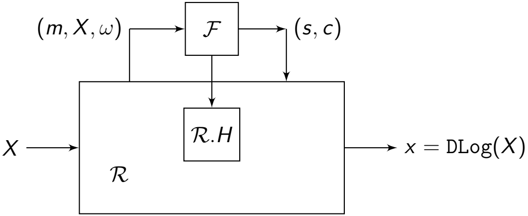
$$s_1 = a + c_1 x \text{ and } s_2 = a + c_2 x \quad \Rightarrow \quad x = \frac{s_1 - s_2}{c_1 - c_2} \mod q$$



Extracting discrete logarithms from a forger

- given a forger $\mathcal{F}$, one can build a reduction $\mathcal{R}$ which solves the DL problem for the public key $X = G^x$ using $\mathcal{F}$ as a black-box
- main idea: have the forger output two forgeries (s_1, c_1) and (s_2, c_2) for the same message m and the same commitment $A = G^a$, so that:

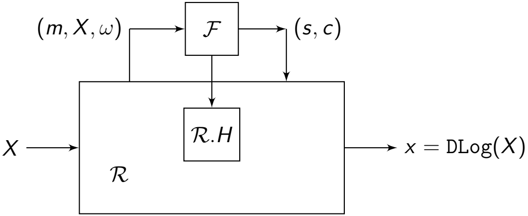
$$s_1 = a + c_1 x \text{ and } s_2 = a + c_2 x \quad \Rightarrow \quad x = \frac{s_1 - s_2}{c_1 - c_2} \mod q$$



Extracting discrete logarithms from a forger

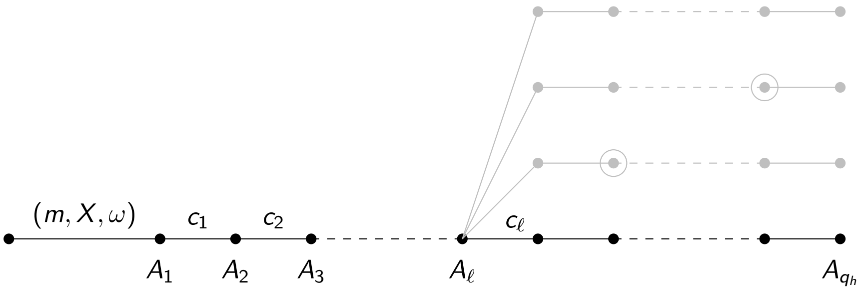
- given a forger $\mathcal{F}$, one can build a reduction $\mathcal{R}$ which solves the DL problem for the public key $X = G^x$ using $\mathcal{F}$ as a black-box
- main idea: have the forger output two forgeries (s_1, c_1) and (s_2, c_2) for the same message m and the same commitment $A = G^a$, so that:

$$s_1 = a + c_1 x \text{ and } s_2 = a + c_2 x \quad \Rightarrow \quad x = \frac{s_1 - s_2}{c_1 - c_2} \mod q$$



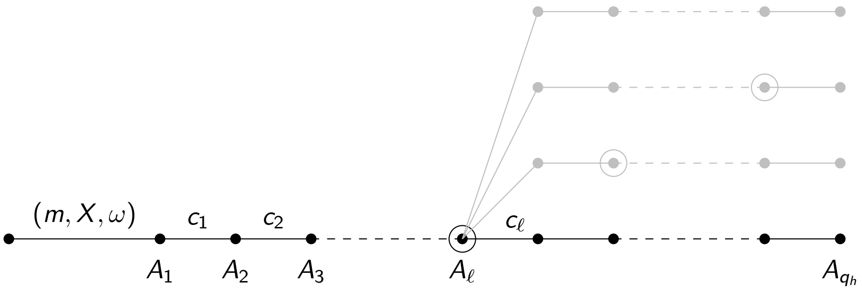
Multiple invocations of the forger: forking

- how does $\mathcal{R}$ obtain two forgeries for the same commitment A?
  $\Rightarrow$ "replay attack"
- run $\mathcal{F}$ until it returns a first forgery for some RO query index $\ell \in [1..q_h]$
- replay the attack up to the forgery point, using new random RO answers from this point
- keep doing this until $\mathcal{F}$ returns a new forgery for the same RO query



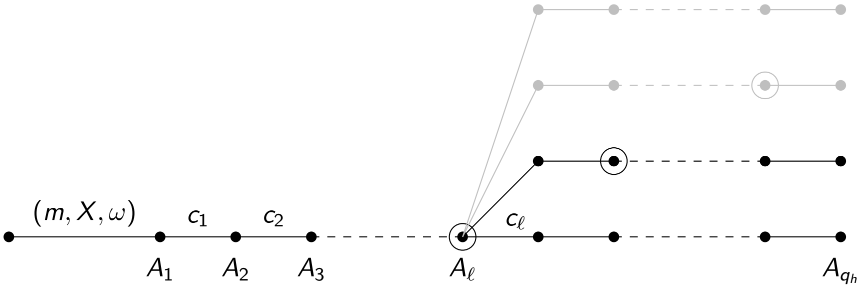
Multiple invocations of the forger: forking

- how does $\mathcal{R}$ obtain two forgeries for the same commitment A?
  $\Rightarrow$ "replay attack"
- run $\mathcal{F}$ until it returns a first forgery for some RO query index $\ell \in [1..q_h]$
- replay the attack up to the forgery point, using new random RO answers from this point
- keep doing this until $\mathcal{F}$ returns a new forgery for the same RO query



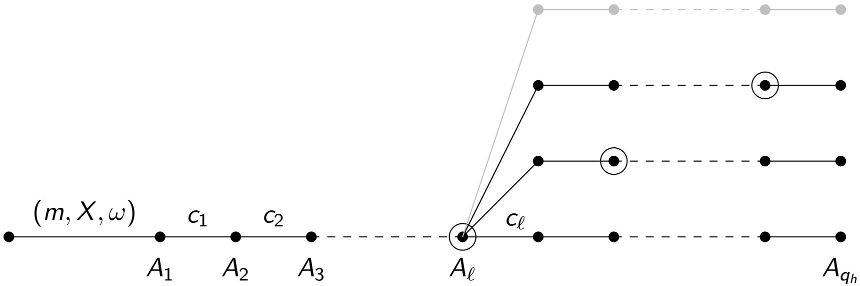
Multiple invocations of the forger: forking

- how does $\mathcal{R}$ obtain two forgeries for the same commitment A?
  $\Rightarrow$ "replay attack"
- run $\mathcal{F}$ until it returns a first forgery for some RO query index $\ell \in [1..q_h]$
- replay the attack up to the forgery point, using new random RO answers from this point
- keep doing this until $\mathcal{F}$ returns a new forgery for the same RO query



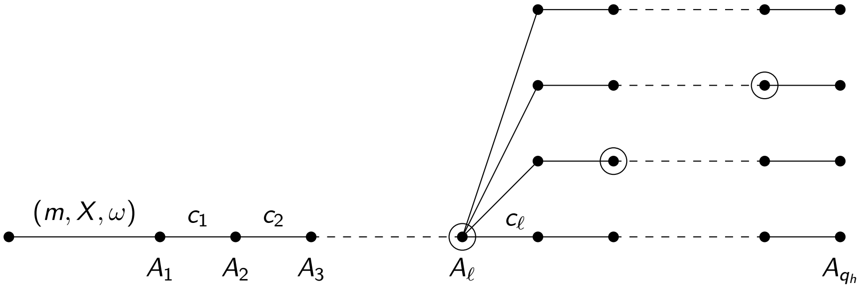
Multiple invocations of the forger: forking

- how does $\mathcal{R}$ obtain two forgeries for the same commitment A?
  $\Rightarrow$ "replay attack"
- run $\mathcal{F}$ until it returns a first forgery for some RO query index $\ell \in [1..q_h]$
- replay the attack up to the forgery point, using new random RO answers from this point
- keep doing this until $\mathcal{F}$ returns a new forgery for the same RO query



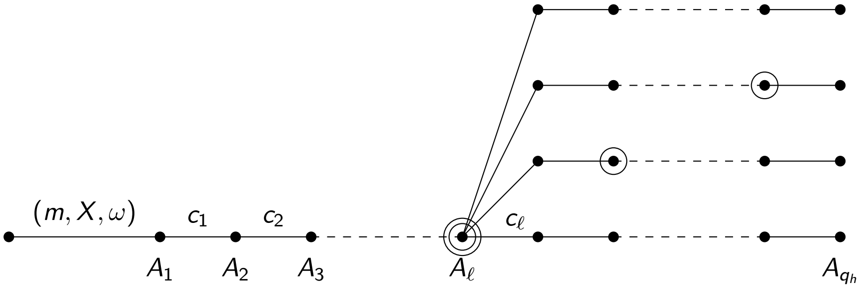
Multiple invocations of the forger: forking

- how does $\mathcal{R}$ obtain two forgeries for the same commitment A?
  $\Rightarrow$ "replay attack"
- run $\mathcal{F}$ until it returns a first forgery for some RO query index $\ell \in [1..q_h]$
- replay the attack up to the forgery point, using new random RO answers from this point
- keep doing this until $\mathcal{F}$ returns a new forgery for the same RO query



Multiple invocations of the forger: forking

- how does $\mathcal{R}$ obtain two forgeries for the same commitment A?
  $\Rightarrow$ "replay attack"
- run $\mathcal{F}$ until it returns a first forgery for some RO query index $\ell \in [1..q_h]$
- replay the attack up to the forgery point, using new random RO answers from this point
- keep doing this until $\mathcal{F}$ returns a new forgery for the same RO query



Success probability of the reduction: the Forking Lemma

- to obtain the first forgery with constant proba.:
  $\Rightarrow$ run the forger $\simeq 1/\varepsilon_F$ times
- to obtain the second forgery with constant proba.:
  $\Rightarrow$ run the forger $\simeq q_h/\varepsilon_F$ times
- total running time $t_R \simeq q_h/\varepsilon_F \times t_F$ for constant success proba.
  $\Rightarrow$ time-to-success ratio of the reduction: $\rho_R \simeq q_h \rho_F$
  $\Rightarrow$ loses a factor q_h
- no matching attack known!
  (best known attack = computing discrete log)

Question

Is there a better reduction with a time-to-success ratio closer to the one of the forger?

Success probability of the reduction: the Forking Lemma

- to obtain the first forgery with constant proba.:
  $\Rightarrow$ run the forger $\simeq 1/\varepsilon_F$ times
- to obtain the second forgery with constant proba.:
  $\Rightarrow$ run the forger $\simeq q_h/\varepsilon_F$ times
- total running time $t_R \simeq q_h/\varepsilon_F \times t_F$ for constant success proba.
  $\Rightarrow$ time-to-success ratio of the reduction: $\rho_R \simeq q_h \rho_F$
  $\Rightarrow$ loses a factor q_h
- no matching attack known!
  (best known attack = computing discrete log)

Question

Is there a better reduction with a time-to-success ratio closer to the one of the forger?

Success probability of the reduction: the Forking Lemma

- to obtain the first forgery with constant proba.:
  $\Rightarrow$ run the forger $\simeq 1/\varepsilon_F$ times
- to obtain the second forgery with constant proba.:
  $\Rightarrow$ run the forger $\simeq q_h/\varepsilon_F$ times
- total running time $t_R \simeq q_h/\varepsilon_F \times t_F$ for constant success proba.
  $\Rightarrow$ time-to-success ratio of the reduction: $\rho_R \simeq q_h \rho_F$
  $\Rightarrow$ loses a factor q_h
- no matching attack known!
  (best known attack = computing discrete log)

Question

Is there a better reduction with a time-to-success ratio closer to the one of the forger?

Success probability of the reduction: the Forking Lemma

- to obtain the first forgery with constant proba.:
  $\Rightarrow$ run the forger $\simeq 1/\varepsilon_F$ times
- to obtain the second forgery with constant proba.:
  $\Rightarrow$ run the forger $\simeq q_h/\varepsilon_F$ times
- total running time $t_R \simeq q_h/\varepsilon_F \times t_F$ for constant success proba.
  $\Rightarrow$ time-to-success ratio of the reduction: $\rho_R \simeq q_h \rho_F$
  $\Rightarrow$ loses a factor q_h
- no matching attack known!
  (best known attack = computing discrete log)

Question

Is there a better reduction with a time-to-success ratio closer to the one of the forger?

Success probability of the reduction: the Forking Lemma

- to obtain the first forgery with constant proba.:
  $\Rightarrow$ run the forger $\simeq 1/\varepsilon_F$ times
- to obtain the second forgery with constant proba.:
  $\Rightarrow$ run the forger $\simeq q_h/\varepsilon_F$ times
- total running time $t_R \simeq q_h/\varepsilon_F \times t_F$ for constant success proba.
  $\Rightarrow$ time-to-success ratio of the reduction: $\rho_R \simeq q_h \rho_F$
  $\Rightarrow$ loses a factor q_h
- no matching attack known!
  (best known attack = computing discrete log)

Question

Is there a better reduction with a time-to-success ratio closer to the one of the forger?

Outline

1. Schnorr Signatures and The Forking Lemma
2. Meta-Reductions
3. Main Result

The concept of meta-reduction

- Boneh and Venkatesan (EC '98) example:
  If there is an (algebraic) reduction $\mathcal{R}$ from factoring to solving the RSA problem with small public exponents, then there is a meta-reduction $\mathcal{M}$ factoring RSA moduli directly (using $\mathcal{R}$)
  $\Rightarrow$ algebraic reductions from factoring to breaking low-RSA exponents cannot exist unless factoring is easy
- here, we will show that an (algebraic) reduction from the Discrete Log (DL) problem to forging Schnorr signatures cannot be tight, unless the One More Discrete Logarithm (OMDL) problem is easy

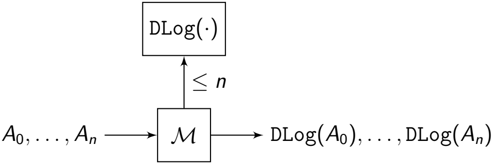
The concept of meta-reduction

- Boneh and Venkatesan (EC '98) example:
  If there is an (algebraic) reduction $\mathcal{R}$ from factoring to solving the RSA problem with small public exponents, then there is a meta-reduction $\mathcal{M}$ factoring RSA moduli directly (using $\mathcal{R}$)
  $\Rightarrow$ algebraic reductions from factoring to breaking low-RSA exponents cannot exist unless factoring is easy
- here, we will show that an (algebraic) reduction from the Discrete Log (DL) problem to forging Schnorr signatures cannot be tight, unless the One More Discrete Logarithm (OMDL) problem is easy

The One More Discrete Logarithm (OMDL) problem

Definition

$\mathcal{M}$ solves the OMDL problem if given $(A_0, A_1, \ldots, A_n) \in_{\mathrm{r}} \mathbb{G}^{n+1}$, it returns the discrete log of all A_i's by making at most n calls to a discrete log oracle $\mathtt{DLog}(\cdot)$.



Restriction to algebraic reductions

Definition

An algorithm $\mathcal{R}$ is algebraic (w.r.t. $\mathbb{G}$) if it only applies group operations on group elements (no bit manipulation, *e.g.* $G \oplus G'$).

Consequence

There exists a procedure `Extract` which, given the group elements $(G_1, \ldots, G_k)$ input to $\mathcal{R}$, $\mathcal{R}$'s code and random tape, and any group element Y output by $\mathcal{R}$, extracts $(\alpha_1, \ldots, \alpha_k)$ such that:

$$Y = G_1^{\alpha_1} \cdots G_k^{\alpha_k}$$

NB: all known reductions for DL-based cryptosystems are algebraic (in particular the reduction of [PS96] for Schnorr signatures)

Restriction to algebraic reductions

Definition

An algorithm $\mathcal{R}$ is algebraic (w.r.t. $\mathbb{G}$) if it only applies group operations on group elements (no bit manipulation, *e.g.* $G \oplus G'$).

Consequence

There exists a procedure `Extract` which, given the group elements $(G_1, \ldots, G_k)$ input to $\mathcal{R}$, $\mathcal{R}$'s code and random tape, and any group element Y output by $\mathcal{R}$, extracts $(\alpha_1, \ldots, \alpha_k)$ such that:

$$Y = G_1^{\alpha_1} \cdots G_k^{\alpha_k}$$

NB: all known reductions for DL-based cryptosystems are algebraic (in particular the reduction of [PS96] for Schnorr signatures)

Restriction to algebraic reductions

Definition

An algorithm $\mathcal{R}$ is algebraic (w.r.t. $\mathbb{G}$) if it only applies group operations on group elements (no bit manipulation, *e.g.* $G \oplus G'$).

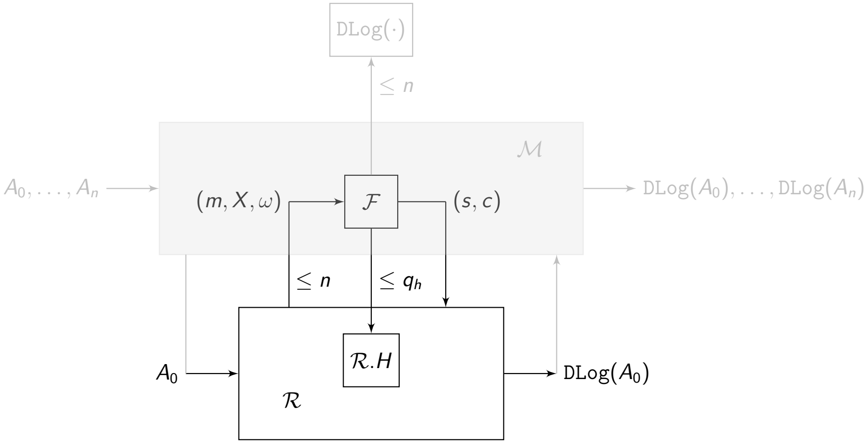
Consequence

There exists a procedure `Extract` which, given the group elements $(G_1, \ldots, G_k)$ input to $\mathcal{R}$, $\mathcal{R}$'s code and random tape, and any group element Y output by $\mathcal{R}$, extracts $(\alpha_1, \ldots, \alpha_k)$ such that:

$$Y = G_1^{\alpha_1} \cdots G_k^{\alpha_k}$$

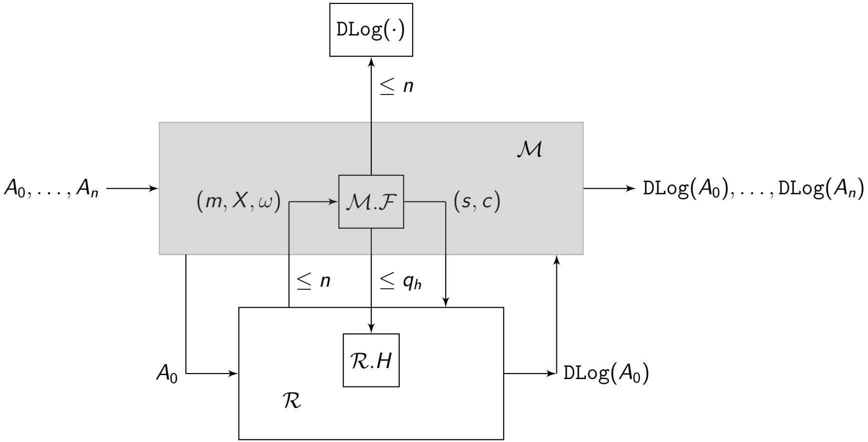
NB: all known reductions for DL-based cryptosystems are algebraic (in particular the reduction of [PS96] for Schnorr signatures)

Meta-reduction: main idea



n=number of times the reduction runs the forger

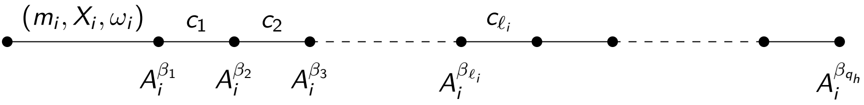
Meta-reduction: main idea



n=number of times the reduction runs the forger

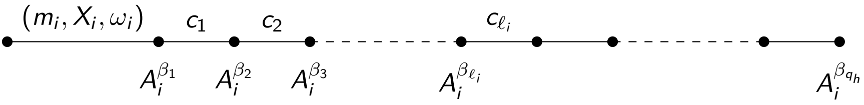
Meta-reduction: the general strategy

- $\mathcal{M}$ receives $(A_0, A_1, \ldots, A_n)$ as input and uses A_0 as input to $\mathcal{R}$
- $\mathcal{M}$ uses A_i, $i = 1, \ldots, n$ during the i-th simulation of the forger to construct q_h commitments $A_i^{\beta_1}, \ldots, A_i^{\beta_{q_h}}$
- for each simulation, $\mathcal{M}$ chooses some forgery index ℓ_i (more on the choice later) and uses its discrete log oracle to forge a signature (s_i, c_i) by querying $s_i = \mathrm{DLog}(A_i^{\beta_{\ell_i}} X_i^{c_{\ell_i}})$
- if the reduction succeeds in returning $a_0 = \mathrm{DLog}(A_0)$, and unless some bad event happens, $\mathcal{M}$ will be able to use a_0 and (s_i, c_i) to compute $a_i = \mathrm{DLog}(A_i)$ for $i = 1, \ldots, n$



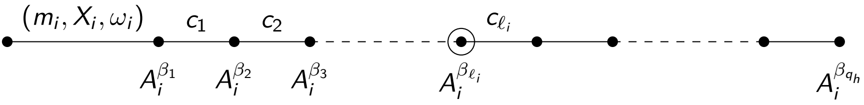
Meta-reduction: the general strategy

- $\mathcal{M}$ receives $(A_0, A_1, \ldots, A_n)$ as input and uses A_0 as input to $\mathcal{R}$
- $\mathcal{M}$ uses A_i, $i = 1, \ldots, n$ during the i-th simulation of the forger to construct q_h commitments $A_i^{\beta_1}, \ldots, A_i^{\beta_{q_h}}$
- for each simulation, $\mathcal{M}$ chooses some forgery index ℓ_i (more on the choice later) and uses its discrete log oracle to forge a signature (s_i, c_i) by querying $s_i = \mathrm{DLog}(A_i^{\beta_{\ell_i}} X_i^{c_{\ell_i}})$
- if the reduction succeeds in returning $a_0 = \mathrm{DLog}(A_0)$, and unless some bad event happens, $\mathcal{M}$ will be able to use a_0 and (s_i, c_i) to compute $a_i = \mathrm{DLog}(A_i)$ for $i = 1, \ldots, n$



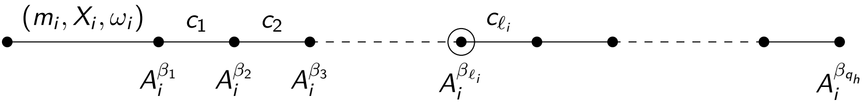
Meta-reduction: the general strategy

- $\mathcal{M}$ receives $(A_0, A_1, \dots, A_n)$ as input and uses A_0 as input to $\mathcal{R}$
- $\mathcal{M}$ uses A_i, $i = 1, \dots, n$ during the i-th simulation of the forger to construct q_h commitments $A_i^{\beta_1}, \dots, A_i^{\beta_{q_h}}$
- for each simulation, $\mathcal{M}$ chooses some forgery index ℓ_i (more on the choice later) and uses its discrete log oracle to forge a signature (s_i, c_i) by querying $s_i = \mathtt{DLog}(A_i^{\beta_{\ell_i}} X_i^{c_{\ell_i}})$
- if the reduction succeeds in returning $a_0 = \mathtt{DLog}(A_0)$, and unless some bad event happens, $\mathcal{M}$ will be able to use a_0 and (s_i, c_i) to compute $a_i = \mathtt{DLog}(A_i)$ for $i = 1, \dots, n$



Meta-reduction: the general strategy

- $\mathcal{M}$ receives $(A_0, A_1, \ldots, A_n)$ as input and uses A_0 as input to $\mathcal{R}$
- $\mathcal{M}$ uses A_i, $i = 1, \ldots, n$ during the i-th simulation of the forger to construct q_h commitments $A_i^{\beta_1}, \ldots, A_i^{\beta_{q_h}}$
- for each simulation, $\mathcal{M}$ chooses some forgery index ℓ_i (more on the choice later) and uses its discrete log oracle to forge a signature (s_i, c_i) by querying $s_i = \mathtt{DLog}(A_i^{\beta_{\ell_i}} X_i^{c_{\ell_i}})$
- if the reduction succeeds in returning $a_0 = \mathtt{DLog}(A_0)$, and unless some bad event happens, $\mathcal{M}$ will be able to use a_0 and (s_i, c_i) to compute $a_i = \mathtt{DLog}(A_i)$ for $i = 1, \ldots, n$



Extraction of $\mathtt{DLog}(A_i)$ by the meta-reduction

- if the simulation of the forger by $\mathcal{M}$ is OK, $\mathcal{R}$ returns $a_0 = \mathtt{DLog}(A_0)$ (with probability $\simeq \varepsilon_R$)
- $\mathcal{M}$ must then use a_0 and the forged signatures (s_i, c_i) to compute $\mathtt{DLog}(A_i)$ for $i = 1, \dots, n$
- the i-th forgery was computed with $s_i = \mathtt{DLog}(A_i^{\beta} X_i^{c_i})$
  $\rightarrow$ computing $\mathtt{DLog}(A_i) \Leftrightarrow$ computing $\mathtt{DLog}(X_i)$
- how can $\mathcal{M}$ retrieve the discrete log of the public keys X_i received from the reduction $\mathcal{R}$?
  $\Rightarrow$ restriction to algebraic reductions
- group elements input to $\mathcal{R}$: G, A_0
- procedure $\mathtt{Extract}$ yields γ_i, γ_i' such that

$$X_i = G^{\gamma_i} A_0^{\gamma_i'} = G^{\gamma_i + a_0 \gamma_i'}$$

Extraction of $\mathtt{DLog}(A_i)$ by the meta-reduction

- if the simulation of the forger by $\mathcal{M}$ is OK, $\mathcal{R}$ returns $a_0 = \mathtt{DLog}(A_0)$ (with probability $\simeq \varepsilon_R$)
- $\mathcal{M}$ must then use a_0 and the forged signatures (s_i, c_i) to compute $\mathtt{DLog}(A_i)$ for $i = 1, \ldots, n$
- the i-th forgery was computed with $s_i = \mathtt{DLog}(A_i^{\beta} X_i^{c_i})$
  $\rightarrow$ computing $\mathtt{DLog}(A_i) \Leftrightarrow$ computing $\mathtt{DLog}(X_i)$
- how can $\mathcal{M}$ retrieve the discrete log of the public keys X_i received from the reduction $\mathcal{R}$?
  $\Rightarrow$ restriction to algebraic reductions
- group elements input to $\mathcal{R}$: G, A_0
- procedure $\mathtt{Extract}$ yields γ_i, γ_i' such that

$$X_i = G^{\gamma_i} A_0^{\gamma_i'} = G^{\gamma_i + a_0 \gamma_i'}$$

Extraction of $\mathtt{DLog}(A_i)$ by the meta-reduction

- if the simulation of the forger by $\mathcal{M}$ is OK, $\mathcal{R}$ returns $a_0 = \mathtt{DLog}(A_0)$ (with probability $\simeq \varepsilon_R$)
- $\mathcal{M}$ must then use a_0 and the forged signatures (s_i, c_i) to compute $\mathtt{DLog}(A_i)$ for $i = 1, \ldots, n$
- the i-th forgery was computed with $s_i = \mathtt{DLog}(A_i^\beta X_i^{c_i})$
  $\rightarrow$ computing $\mathtt{DLog}(A_i) \Leftrightarrow$ computing $\mathtt{DLog}(X_i)$
- how can $\mathcal{M}$ retrieve the discrete log of the public keys X_i received from the reduction $\mathcal{R}$?
  $\Rightarrow$ restriction to algebraic reductions
- group elements input to $\mathcal{R}$: G, A_0
- procedure $\mathtt{Extract}$ yields γ_i, γ_i' such that

$$X_i = G^{\gamma_i} A_0^{\gamma_i'} = G^{\gamma_i + a_0 \gamma_i'}$$

Extraction of $\mathtt{DLog}(A_i)$ by the meta-reduction

- if the simulation of the forger by $\mathcal{M}$ is OK, $\mathcal{R}$ returns $a_0 = \mathtt{DLog}(A_0)$ (with probability $\simeq \varepsilon_R$)
- $\mathcal{M}$ must then use a_0 and the forged signatures (s_i, c_i) to compute $\mathtt{DLog}(A_i)$ for $i = 1, \ldots, n$
- the i-th forgery was computed with $s_i = \mathtt{DLog}(A_i^{\beta} X_i^{c_i})$
  $\rightarrow$ computing $\mathtt{DLog}(A_i) \Leftrightarrow$ computing $\mathtt{DLog}(X_i)$
- how can $\mathcal{M}$ retrieve the discrete log of the public keys X_i received from the reduction $\mathcal{R}$?
  $\Rightarrow$ restriction to algebraic reductions
- group elements input to $\mathcal{R}$: G, A_0
- procedure $\mathtt{Extract}$ yields γ_i, γ_i' such that

$$X_i = G^{\gamma_i} A_0^{\gamma_i'} = G^{\gamma_i + a_0 \gamma_i'}$$

Extraction of $\mathtt{DLog}(A_i)$ by the meta-reduction

- if the simulation of the forger by $\mathcal{M}$ is OK, $\mathcal{R}$ returns $a_0 = \mathtt{DLog}(A_0)$ (with probability $\simeq \varepsilon_R$)
- $\mathcal{M}$ must then use a_0 and the forged signatures (s_i, c_i) to compute $\mathtt{DLog}(A_i)$ for $i = 1, \ldots, n$
- the i-th forgery was computed with $s_i = \mathtt{DLog}(A_i^{\beta} X_i^{c_i})$
  $\rightarrow$ computing $\mathtt{DLog}(A_i) \Leftrightarrow$ computing $\mathtt{DLog}(X_i)$
- how can $\mathcal{M}$ retrieve the discrete log of the public keys X_i received from the reduction $\mathcal{R}$?
  $\Rightarrow$ restriction to algebraic reductions
- group elements input to $\mathcal{R}$: G, A_0
- procedure $\mathtt{Extract}$ yields γ_i, γ_i' such that

$$X_i = G^{\gamma_i} A_0^{\gamma_i'} = G^{\gamma_i + a_0 \gamma_i'}$$

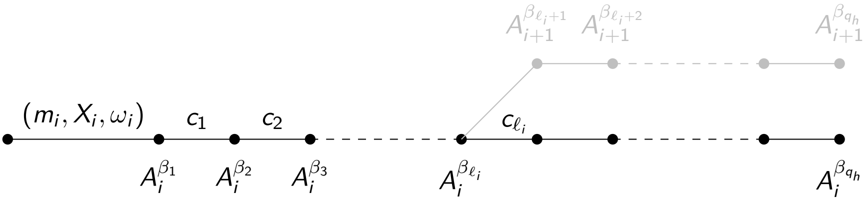
Extraction of $\mathtt{DLog}(A_i)$ by the meta-reduction

- if the simulation of the forger by $\mathcal{M}$ is OK, $\mathcal{R}$ returns $a_0 = \mathtt{DLog}(A_0)$ (with probability $\simeq \varepsilon_R$)
- $\mathcal{M}$ must then use a_0 and the forged signatures (s_i, c_i) to compute $\mathtt{DLog}(A_i)$ for $i = 1, \ldots, n$
- the i-th forgery was computed with $s_i = \mathtt{DLog}(A_i^\beta X_i^{c_i})$
  $\rightarrow$ computing $\mathtt{DLog}(A_i) \Leftrightarrow$ computing $\mathtt{DLog}(X_i)$
- how can $\mathcal{M}$ retrieve the discrete log of the public keys X_i received from the reduction $\mathcal{R}$?
  $\Rightarrow$ restriction to algebraic reductions
- group elements input to $\mathcal{R}$: G, A_0
- procedure $\mathtt{Extract}$ yields γ_i, γ_i' such that

$$X_i = G^{\gamma_i} A_0^{\gamma_i'} = G^{\gamma_i + a_0 \gamma_i'}$$

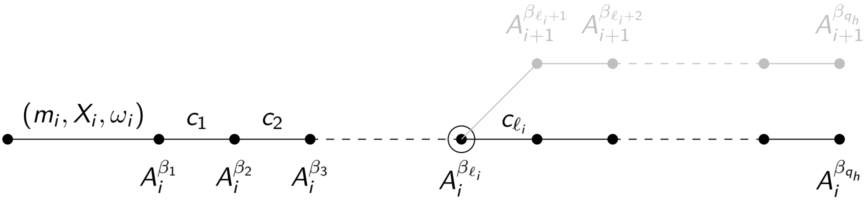
A bad event which makes the meta-reduction fail

- two simulations may share some common history (under control of $\mathcal{R}$!) as in the Forking Lemma
- $\mathcal{M}$ fails if it forges two signatures for the same commitment because it will make a useless call to $\mathtt{DLog}(\cdot)$ $\rightarrow$ event Bad happens
- NB: this is exactly the event which makes the reduction succeed in the Forking Lemma
- unless $\Pr[\mathsf{Bad}] \simeq 1$, we get a contradiction since otherwise $\mathcal{M}$ is an efficient and successful algorithm for the OMDL problem



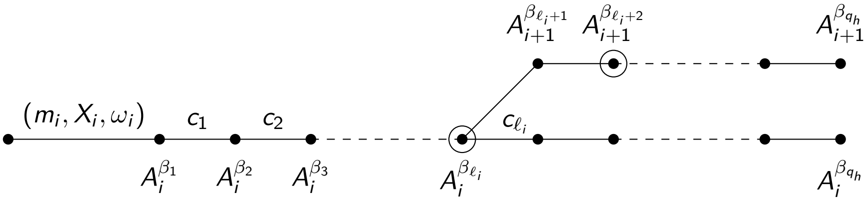
A bad event which makes the meta-reduction fail

- two simulations may share some common history (under control of $\mathcal{R}$!) as in the Forking Lemma
- $\mathcal{M}$ fails if it forges two signatures for the same commitment because it will make a useless call to $\mathtt{DLog}(\cdot)$ $\rightarrow$ event Bad happens
- NB: this is exactly the event which makes the reduction succeed in the Forking Lemma
- unless $\Pr[\mathrm{Bad}] \simeq 1$, we get a contradiction since otherwise $\mathcal{M}$ is an efficient and successful algorithm for the OMDL problem



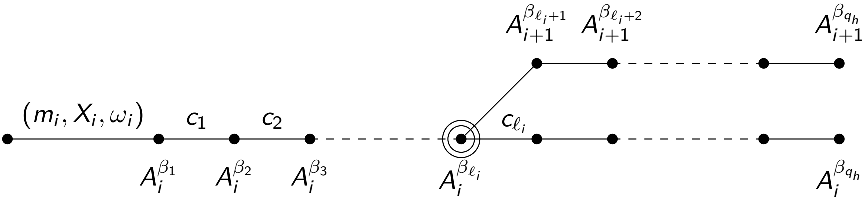
A bad event which makes the meta-reduction fail

- two simulations may share some common history (under control of $\mathcal{R}$!) as in the Forking Lemma
- $\mathcal{M}$ fails if it forges two signatures for the same commitment because it will make a useless call to $\mathtt{DLog}(\cdot)$ $\rightarrow$ event Bad happens
- NB: this is exactly the event which makes the reduction succeed in the Forking Lemma
- unless $\Pr[\mathrm{Bad}] \simeq 1$, we get a contradiction since otherwise $\mathcal{M}$ is an efficient and successful algorithm for the OMDL problem



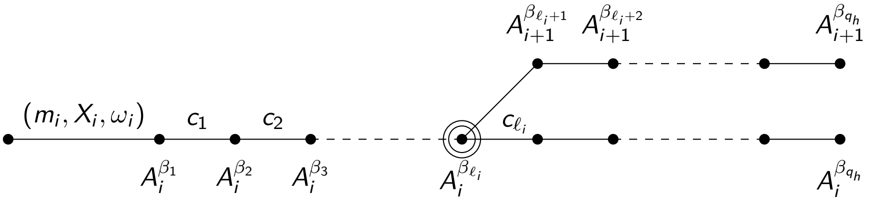
A bad event which makes the meta-reduction fail

- two simulations may share some common history (under control of $\mathcal{R}$!) as in the Forking Lemma
- $\mathcal{M}$ fails if it forges two signatures for the same commitment because it will make a useless call to $\mathtt{DLog}(\cdot)$ $\rightarrow$ event $\mathtt{Bad}$ happens
- NB: this is exactly the event which makes the reduction succeed in the Forking Lemma
- unless $\Pr[\mathtt{Bad}] \simeq 1$, we get a contradiction since otherwise $\mathcal{M}$ is an efficient and successful algorithm for the OMDL problem



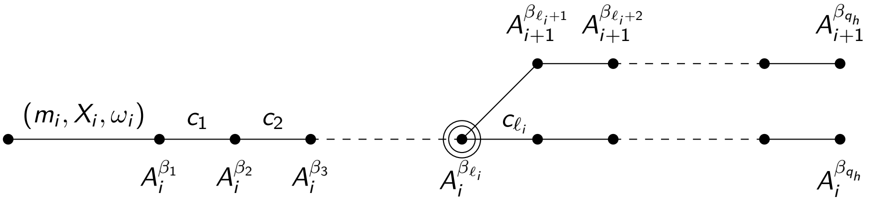
A bad event which makes the meta-reduction fail

- two simulations may share some common history (under control of $\mathcal{R}$!) as in the Forking Lemma
- $\mathcal{M}$ fails if it forges two signatures for the same commitment because it will make a useless call to $\mathtt{DLog}(\cdot)$ $\rightarrow$ event $\mathtt{Bad}$ happens
- NB: this is exactly the event which makes the reduction succeed in the Forking Lemma
- unless $\Pr[\mathtt{Bad}] \simeq 1$, we get a contradiction since otherwise $\mathcal{M}$ is an efficient and successful algorithm for the OMDL problem



A bad event which makes the meta-reduction fail

- two simulations may share some common history (under control of $\mathcal{R}$!) as in the Forking Lemma
- $\mathcal{M}$ fails if it forges two signatures for the same commitment because it will make a useless call to $\mathtt{DLog}(\cdot)$ $\rightarrow$ event $\mathtt{Bad}$ happens
- NB: this is exactly the event which makes the reduction succeed in the Forking Lemma
- unless $\Pr[\mathtt{Bad}] \simeq 1$, we get a contradiction since otherwise $\mathcal{M}$ is an efficient and successful algorithm for the OMDL problem



Simulation of the forger: choice of the forgery index

- how should the meta-reduction choose the forgery index ℓ_i for the i-th execution?
- cannot choose $\ell_1 = 1$, $\ell_2 = 2$, etc. (the reduction would "notice" that a simulation is ongoing)
- natural choice: draw ℓ_i uniformly at random in $[1..q_h]$ independently for each execution $i = 1, \ldots, n$
- this is what was done in previous work [PV05,GBL08]
- straightforward analysis [PV05]:

$$\Pr[\mathrm{Bad}] \simeq \frac{n^2}{q_h} \quad \Rightarrow \quad n \simeq q_h^{1/2} \text{ for } \Pr[\mathrm{Bad}] \simeq 1$$

- more careful analysis [GBL08]:

$$\Pr[\mathrm{Bad}] \simeq \frac{n^{3/2}}{q_h} \quad \Rightarrow \quad n \simeq q_h^{2/3} \text{ for } \Pr[\mathrm{Bad}] \simeq 1$$

Simulation of the forger: choice of the forgery index

- how should the meta-reduction choose the forgery index ℓ_i for the i-th execution?
- cannot choose $\ell_1 = 1$, $\ell_2 = 2$, etc. (the reduction would "notice" that a simulation is ongoing)
- natural choice: draw ℓ_i uniformly at random in $[1..q_h]$ independently for each execution $i = 1, \ldots, n$
- this is what was done in previous work [PV05,GBL08]
- straightforward analysis [PV05]:

$$\Pr[\mathrm{Bad}] \simeq \frac{n^2}{q_h} \quad \Rightarrow \quad n \simeq q_h^{1/2} \text{ for } \Pr[\mathrm{Bad}] \simeq 1$$

- more careful analysis [GBL08]:

$$\Pr[\mathrm{Bad}] \simeq \frac{n^{3/2}}{q_h} \quad \Rightarrow \quad n \simeq q_h^{2/3} \text{ for } \Pr[\mathrm{Bad}] \simeq 1$$

Simulation of the forger: choice of the forgery index

- how should the meta-reduction choose the forgery index ℓ_i for the i-th execution?
- cannot choose $\ell_1 = 1$, $\ell_2 = 2$, etc. (the reduction would "notice" that a simulation is ongoing)
- natural choice: draw ℓ_i uniformly at random in $[1..q_h]$ independently for each execution $i = 1, \ldots, n$
- this is what was done in previous work [PV05,GBL08]
- straightforward analysis [PV05]:

$$\Pr[\mathrm{Bad}] \simeq \frac{n^2}{q_h} \quad \Rightarrow \quad n \simeq q_h^{1/2} \text{ for } \Pr[\mathrm{Bad}] \simeq 1$$

- more careful analysis [GBL08]:

$$\Pr[\mathrm{Bad}] \simeq \frac{n^{3/2}}{q_h} \quad \Rightarrow \quad n \simeq q_h^{2/3} \text{ for } \Pr[\mathrm{Bad}] \simeq 1$$

Simulation of the forger: choice of the forgery index

- how should the meta-reduction choose the forgery index ℓ_i for the i-th execution?
- cannot choose $\ell_1 = 1$, $\ell_2 = 2$, etc. (the reduction would "notice" that a simulation is ongoing)
- natural choice: draw ℓ_i uniformly at random in $[1..q_h]$ independently for each execution $i = 1, \ldots, n$
- this is what was done in previous work [PV05,GBL08]
- straightforward analysis [PV05]:

$$\Pr[\text{Bad}] \simeq \frac{n^2}{q_h} \quad \Rightarrow \quad n \simeq q_h^{1/2} \text{ for } \Pr[\text{Bad}] \simeq 1$$

- more careful analysis [GBL08]:

$$\Pr[\text{Bad}] \simeq \frac{n^{3/2}}{q_h} \quad \Rightarrow \quad n \simeq q_h^{2/3} \text{ for } \Pr[\text{Bad}] \simeq 1$$

Simulation of the forger: choice of the forgery index

- how should the meta-reduction choose the forgery index ℓ_i for the i-th execution?
- cannot choose $\ell_1 = 1$, $\ell_2 = 2$, etc. (the reduction would "notice" that a simulation is ongoing)
- natural choice: draw ℓ_i uniformly at random in $[1..q_h]$ independently for each execution $i = 1, \ldots, n$
- this is what was done in previous work [PV05,GBL08]
- straightforward analysis [PV05]:

$$\Pr[\mathrm{Bad}] \simeq \frac{n^2}{q_h} \quad \Rightarrow \quad n \simeq q_h^{1/2} \text{ for } \Pr[\mathrm{Bad}] \simeq 1$$

- more careful analysis [GBL08]:

$$\Pr[\mathrm{Bad}] \simeq \frac{n^{3/2}}{q_h} \quad \Rightarrow \quad n \simeq q_h^{2/3} \text{ for } \Pr[\mathrm{Bad}] \simeq 1$$

Simulation of the forger: choice of the forgery index

- how should the meta-reduction choose the forgery index ℓ_i for the i-th execution?
- cannot choose $\ell_1 = 1$, $\ell_2 = 2$, etc. (the reduction would "notice" that a simulation is ongoing)
- natural choice: draw ℓ_i uniformly at random in $[1..q_h]$ independently for each execution $i = 1, \ldots, n$
- this is what was done in previous work [PV05,GBL08]
- straightforward analysis [PV05]:

$$\Pr[\mathrm{Bad}] \simeq \frac{n^2}{q_h} \quad \Rightarrow \quad n \simeq q_h^{1/2} \text{ for } \Pr[\mathrm{Bad}] \simeq 1$$

- more careful analysis [GBL08]:

$$\Pr[\mathrm{Bad}] \simeq \frac{n^{3/2}}{q_h} \quad \Rightarrow \quad n \simeq q_h^{2/3} \text{ for } \Pr[\mathrm{Bad}] \simeq 1$$

Outline

1 Schnorr Signatures and The Forking Lemma

2 Meta-Reductions

3 Main Result

Main theorem

Theorem

Any algebraic reduction from the DL problem to forging Schnorr signatures must lose a factor q_h in its time-to-success ratio, assuming the OMDL problem is hard.

- for strictly bounded adversaries, factor $f(\varepsilon_F)q_h$ with $f(\varepsilon_F)$ close to 1 as long as $\varepsilon_F < 0.9$
- for expected-time and queries adversaries, factor q_h independently of ε_F
- proof: new meta-reduction (crucial modification = choice of the forgery index ℓ for the simulated forger)

Main theorem

Theorem

Any algebraic reduction from the DL problem to forging Schnorr signatures must lose a factor q_h in its time-to-success ratio, assuming the OMDL problem is hard.

- for strictly bounded adversaries, factor $f(\varepsilon_F)q_h$ with $f(\varepsilon_F)$ close to 1 as long as $\varepsilon_F < 0.9$
- for expected-time and queries adversaries, factor q_h independently of ε_F
- proof: new meta-reduction (crucial modification = choice of the forgery index ℓ for the simulated forger)

Main theorem

Theorem

Any algebraic reduction from the DL problem to forging Schnorr signatures must lose a factor q_h in its time-to-success ratio, assuming the OMDL problem is hard.

- for strictly bounded adversaries, factor $f(\varepsilon_F)q_h$ with $f(\varepsilon_F)$ close to 1 as long as $\varepsilon_F < 0.9$
- for expected-time and queries adversaries, factor q_h independently of ε_F
- proof: new meta-reduction (crucial modification = choice of the forgery index ℓ for the simulated forger)

Main theorem

Theorem

Any algebraic reduction from the DL problem to forging Schnorr signatures must lose a factor q_h in its time-to-success ratio, assuming the OMDL problem is hard.

- for strictly bounded adversaries, factor $f(\varepsilon_F)q_h$ with $f(\varepsilon_F)$ close to 1 as long as $\varepsilon_F < 0.9$
- for expected-time and queries adversaries, factor q_h independently of ε_F
- proof: new meta-reduction (crucial modification $=$ choice of the forgery index ℓ for the simulated forger)

A thought experiment

- consider the following hypothetic forger $\mathcal{F}$:
- $\mathbb{G}$ is partitioned into two sets:
  - Γ_{good} of size $\mu|\mathbb{G}|$: $\mathcal{F}$ can compute discrete logs efficiently for this set
  - Γ_{bad} of size $(1-\mu)|\mathbb{G}|$: $\mathcal{F}$ cannot compute discrete logs for this set
- to forge a signature for m, $\mathcal{F}$ makes arbitrary RO queries $\boldsymbol{H}(m, A_i) = c_i$ and returns a forgery for the first query such that $A_i X^{c_i} \in \Gamma_{\text{good}}$ (or fails to forge if there is no such query)
- success probability of $\mathcal{F}$ if it makes q_h RO queries:
  - for each RO query, $A_i X^{c_i}$ is unif. random in $\mathbb{G}$
    $\Rightarrow A_i X^{c_i} \in \Gamma_{\text{good}}$ with proba. μ
  - hence $\boxed{\varepsilon_F = 1 - (1-\mu)^{q_h}}$
- we will call such a $\mathcal{F}$ a μ-good forger

A thought experiment

- consider the following hypothetic forger $\mathcal{F}$:
- $\mathbb{G}$ is partitioned into two sets:
  - Γ_{good} of size $\mu|\mathbb{G}|$: $\mathcal{F}$ can compute discrete logs efficiently for this set
  - Γ_{bad} of size $(1-\mu)|\mathbb{G}|$: $\mathcal{F}$ cannot compute discrete logs for this set
- to forge a signature for m, $\mathcal{F}$ makes arbitrary RO queries $\boldsymbol{H}(m, A_i) = c_i$ and returns a forgery for the first query such that $A_i X^{c_i} \in \Gamma_{\text{good}}$ (or fails to forge if there is no such query)
- success probability of $\mathcal{F}$ if it makes q_h RO queries:
  - for each RO query, $A_i X^{c_i}$ is unif. random in $\mathbb{G}$
    $\Rightarrow A_i X^{c_i} \in \Gamma_{\text{good}}$ with proba. μ
  - hence $\boxed{\varepsilon_F = 1 - (1-\mu)^{q_h}}$
- we will call such a $\mathcal{F}$ a μ-good forger

A thought experiment

- consider the following hypothetic forger $\mathcal{F}$:
- $\mathbb{G}$ is partitioned into two sets:
  - Γ_{good} of size $\mu|\mathbb{G}|$: $\mathcal{F}$ can compute discrete logs efficiently for this set
  - Γ_{bad} of size $(1-\mu)|\mathbb{G}|$: $\mathcal{F}$ cannot compute discrete logs for this set
- to forge a signature for m, $\mathcal{F}$ makes arbitrary RO queries $\boldsymbol{H}(m, A_i) = c_i$ and returns a forgery for the first query such that $A_i X^{c_i} \in \Gamma_{\mathrm{good}}$ (or fails to forge if there is no such query)
- success probability of $\mathcal{F}$ if it makes q_h RO queries:
  - for each RO query, $A_i X^{c_i}$ is unif. random in $\mathbb{G}$ $\Rightarrow A_i X^{c_i} \in \Gamma_{\mathrm{good}}$ with proba. μ
  - hence $\boxed{\varepsilon_F = 1-(1-\mu)^{q_h}}$
- we will call such a $\mathcal{F}$ a μ-good forger

A thought experiment

- consider the following hypothetic forger $\mathcal{F}$:
- $\mathbb{G}$ is partitioned into two sets:
  - Γ_{good} of size $\mu|\mathbb{G}|$: $\mathcal{F}$ can compute discrete logs efficiently for this set
  - Γ_{bad} of size $(1-\mu)|\mathbb{G}|$: $\mathcal{F}$ cannot compute discrete logs for this set
- to forge a signature for m, $\mathcal{F}$ makes arbitrary RO queries $\boldsymbol{H}(m, A_i) = c_i$ and returns a forgery for the first query such that $A_i X^{c_i} \in \Gamma_{\mathrm{good}}$ (or fails to forge if there is no such query)
- success probability of $\mathcal{F}$ if it makes q_h RO queries:
  - for each RO query, $A_i X^{c_i}$ is unif. random in $\mathbb{G}$ $\Rightarrow A_i X^{c_i} \in \Gamma_{\mathrm{good}}$ with proba. μ
  - hence $\boxed{\varepsilon_F = 1 - (1-\mu)^{q_h}}$
- we will call such a $\mathcal{F}$ a μ-good forger

A thought experiment

- consider the following hypothetic forger $\mathcal{F}$:
- $\mathbb{G}$ is partitioned into two sets:
  - Γ_{good} of size $\mu|\mathbb{G}|$: $\mathcal{F}$ can compute discrete logs efficiently for this set
  - Γ_{bad} of size $(1-\mu)|\mathbb{G}|$: $\mathcal{F}$ cannot compute discrete logs for this set
- to forge a signature for m, $\mathcal{F}$ makes arbitrary RO queries $\boldsymbol{H}(m, A_i) = c_i$ and returns a forgery for the first query such that $A_i X^{c_i} \in \Gamma_{\text{good}}$ (or fails to forge if there is no such query)
- success probability of $\mathcal{F}$ if it makes q_h RO queries:
  - for each RO query, $A_i X^{c_i}$ is unif. random in $\mathbb{G}$ $\Rightarrow A_i X^{c_i} \in \Gamma_{\text{good}}$ with proba. μ
  - hence $\boxed{\varepsilon_F = 1 - (1-\mu)^{q_h}}$
- we will call such a $\mathcal{F}$ a μ-good forger

A thought experiment

- consider the following hypothetic forger $\mathcal{F}$:
- $\mathbb{G}$ is partitioned into two sets:
  - Γ_{good} of size $\mu|\mathbb{G}|$: $\mathcal{F}$ can compute discrete logs efficiently for this set
  - Γ_{bad} of size $(1-\mu)|\mathbb{G}|$: $\mathcal{F}$ cannot compute discrete logs for this set
- to forge a signature for m, $\mathcal{F}$ makes arbitrary RO queries $\boldsymbol{H}(m, A_i) = c_i$ and returns a forgery for the first query such that $A_i X^{c_i} \in \Gamma_{\mathrm{good}}$ (or fails to forge if there is no such query)
- success probability of $\mathcal{F}$ if it makes q_h RO queries:
  - for each RO query, $A_i X^{c_i}$ is unif. random in $\mathbb{G}$ $\Rightarrow A_i X^{c_i} \in \Gamma_{\mathrm{good}}$ with proba. μ
  - hence $\boxed{\varepsilon_F = 1 - (1-\mu)^{q_h}}$
- we will call such a $\mathcal{F}$ a μ-good forger

A thought experiment

- consider the following hypothetic forger $\mathcal{F}$:
- $\mathbb{G}$ is partitioned into two sets:
  - Γ_{good} of size $\mu|\mathbb{G}|$: $\mathcal{F}$ can compute discrete logs efficiently for this set
  - Γ_{bad} of size $(1-\mu)|\mathbb{G}|$: $\mathcal{F}$ cannot compute discrete logs for this set
- to forge a signature for m, $\mathcal{F}$ makes arbitrary RO queries $\boldsymbol{H}(m, A_i) = c_i$ and returns a forgery for the first query such that $A_i X^{c_i} \in \Gamma_{\mathrm{good}}$ (or fails to forge if there is no such query)
- success probability of $\mathcal{F}$ if it makes q_h RO queries:
  - for each RO query, $A_i X^{c_i}$ is unif. random in $\mathbb{G}$
    $\Rightarrow A_i X^{c_i} \in \Gamma_{\mathrm{good}}$ with proba. μ
  - hence $\boxed{\varepsilon_F = 1 - (1-\mu)^{q_h}}$
- we will call such a $\mathcal{F}$ a μ-good forger

A thought experiment

- consider the following hypothetic forger $\mathcal{F}$:
- $\mathbb{G}$ is partitioned into two sets:
  - Γ_{good} of size $\mu|\mathbb{G}|$: $\mathcal{F}$ can compute discrete logs efficiently for this set
  - Γ_{bad} of size $(1-\mu)|\mathbb{G}|$: $\mathcal{F}$ cannot compute discrete logs for this set
- to forge a signature for m, $\mathcal{F}$ makes arbitrary RO queries $\boldsymbol{H}(m, A_i) = c_i$ and returns a forgery for the first query such that $A_i X^{c_i} \in \Gamma_{\mathrm{good}}$ (or fails to forge if there is no such query)
- success probability of $\mathcal{F}$ if it makes q_h RO queries:
  - for each RO query, $A_i X^{c_i}$ is unif. random in $\mathbb{G}$ $\Rightarrow A_i X^{c_i} \in \Gamma_{\mathrm{good}}$ with proba. μ
  - hence $\boxed{\varepsilon_F = 1 - (1-\mu)^{q_h}}$
- we will call such a $\mathcal{F}$ a μ-good forger

A thought experiment

- consider the following hypothetic forger $\mathcal{F}$:
- $\mathbb{G}$ is partitioned into two sets:
  - Γ_{good} of size $\mu|\mathbb{G}|$: $\mathcal{F}$ can compute discrete logs efficiently for this set
  - Γ_{bad} of size $(1-\mu)|\mathbb{G}|$: $\mathcal{F}$ cannot compute discrete logs for this set
- to forge a signature for m, $\mathcal{F}$ makes arbitrary RO queries $\boldsymbol{H}(m, A_i) = c_i$ and returns a forgery for the first query such that $A_i X^{c_i} \in \Gamma_{\text{good}}$ (or fails to forge if there is no such query)
- success probability of $\mathcal{F}$ if it makes q_h RO queries:
  - for each RO query, $A_i X^{c_i}$ is unif. random in $\mathbb{G}$ $\Rightarrow A_i X^{c_i} \in \Gamma_{\text{good}}$ with proba. μ
  - hence $\boxed{\varepsilon_F = 1 - (1-\mu)^{q_h}}$
- we will call such a $\mathcal{F}$ a μ-good forger

The new meta-reduction

- we define a meta-reduction $\mathcal{M}$ which simulates a μ-good forger
- $\mathcal{M}$ builds Γ_{good} and Γ_{bad} dynamically and randomly during the simulation as follows:
  - for each RO query $\mathcal{R}.H(m, A) = c$, define $Z = AX^c$
  - if $Z \notin \Gamma_{\text{good}} \cup \Gamma_{\text{bad}}$, draw a random coin δ_Z with

$$\Pr[\delta_Z = 1] = \mu \text{ and } \Pr[\delta_Z = 0] = 1 - \mu$$

    and add Z to Γ_{good} if $\delta_Z = 1$ or to Γ_{bad} if $\delta_Z = 0$.
- discrete logs of elements of Γ_{good} are obtained thanks to the discrete log oracle of $\mathcal{M}$
- the forgery index ℓ_i is distributed according to a (truncated) geometric distribution of parameter μ

The new meta-reduction

- we define a meta-reduction $\mathcal{M}$ which simulates a μ-good forger
- $\mathcal{M}$ builds Γ_{good} and Γ_{bad} dynamically and randomly during the simulation as follows:
  - for each RO query $\mathcal{R}.H(m, A) = c$, define $Z = AX^c$
  - if $Z \notin \Gamma_{\text{good}} \cup \Gamma_{\text{bad}}$, draw a random coin δ_Z with

$$\Pr[\delta_Z = 1] = \mu \text{ and } \Pr[\delta_Z = 0] = 1 - \mu$$

    and add Z to Γ_{good} if $\delta_Z = 1$ or to Γ_{bad} if $\delta_Z = 0$.
- discrete logs of elements of Γ_{good} are obtained thanks to the discrete log oracle of $\mathcal{M}$
- the forgery index ℓ_i is distributed according to a (truncated) geometric distribution of parameter μ

The new meta-reduction

- we define a meta-reduction $\mathcal{M}$ which simulates a μ-good forger
- $\mathcal{M}$ builds Γ_{good} and Γ_{bad} dynamically and randomly during the simulation as follows:
  - for each RO query $\mathcal{R}.H(m, A) = c$, define $Z = AX^c$
  - if $Z \notin \Gamma_{\text{good}} \cup \Gamma_{\text{bad}}$, draw a random coin δ_Z with

    $$\Pr[\delta_Z = 1] = \mu \text{ and } \Pr[\delta_Z = 0] = 1 - \mu$$

    and add Z to Γ_{good} if $\delta_Z = 1$ or to Γ_{bad} if $\delta_Z = 0$.
- discrete logs of elements of Γ_{good} are obtained thanks to the discrete log oracle of $\mathcal{M}$
- the forgery index ℓ_i is distributed according to a (truncated) geometric distribution of parameter μ

The new meta-reduction

- we define a meta-reduction $\mathcal{M}$ which simulates a μ-good forger
- $\mathcal{M}$ builds Γ_{good} and Γ_{bad} dynamically and randomly during the simulation as follows:
  - for each RO query $\mathcal{R}.H(m, A) = c$, define $Z = AX^c$
  - if $Z \notin \Gamma_{\mathrm{good}} \cup \Gamma_{\mathrm{bad}}$, draw a random coin δ_Z with

$$\Pr[\delta_Z = 1] = \mu \text{ and } \Pr[\delta_Z = 0] = 1 - \mu$$

  and add Z to Γ_{good} if $\delta_Z = 1$ or to Γ_{bad} if $\delta_Z = 0$.
- discrete logs of elements of Γ_{good} are obtained thanks to the discrete log oracle of $\mathcal{M}$
- the forgery index ℓ_i is distributed according to a (truncated) geometric distribution of parameter μ

The new meta-reduction

- we define a meta-reduction $\mathcal{M}$ which simulates a μ-good forger
- $\mathcal{M}$ builds Γ_{good} and Γ_{bad} dynamically and randomly during the simulation as follows:
  - for each RO query $\mathcal{R}.H(m, A) = c$, define $Z = AX^c$
  - if $Z \notin \Gamma_{\mathrm{good}} \cup \Gamma_{\mathrm{bad}}$, draw a random coin δ_Z with

$$\Pr[\delta_Z = 1] = \mu \text{ and } \Pr[\delta_Z = 0] = 1 - \mu$$

  and add Z to Γ_{good} if $\delta_Z = 1$ or to Γ_{bad} if $\delta_Z = 0$.
- discrete logs of elements of Γ_{good} are obtained thanks to the discrete log oracle of $\mathcal{M}$
- the forgery index ℓ_i is distributed according to a (truncated) geometric distribution of parameter μ

The new meta-reduction

- we define a meta-reduction $\mathcal{M}$ which simulates a μ-good forger
- $\mathcal{M}$ builds Γ_{good} and Γ_{bad} dynamically and randomly during the simulation as follows:
  - for each RO query $\mathcal{R}.H(m, A) = c$, define $Z = AX^c$
  - if $Z \notin \Gamma_{\text{good}} \cup \Gamma_{\text{bad}}$, draw a random coin δ_Z with

$$\Pr[\delta_Z = 1] = \mu \text{ and } \Pr[\delta_Z = 0] = 1 - \mu$$

  and add Z to Γ_{good} if $\delta_Z = 1$ or to Γ_{bad} if $\delta_Z = 0$.
- discrete logs of elements of Γ_{good} are obtained thanks to the discrete log oracle of $\mathcal{M}$
- the forgery index ℓ_i is distributed according to a (truncated) geometric distribution of parameter μ

$\mathcal{M}$ "almost always" simulates a μ-good forger

- the size of Γ_{good} defined by $\mathcal{M}$ follows a binomial distribution of parameters $(|G|, \mu)$
  $\Rightarrow$ by a Chernoff bound, $|\Gamma_{\mathrm{good}}| \simeq \mu|\mathbb{G}|$ with overwhelming probability
- in that case, the success probability of the simulated forger satisfies:

$$\varepsilon_F = 1 - (1-\mu)^{q_h}$$

- by setting μ appropriately, $\mathcal{M}$ can simulate a forger achieving the required success probability ε_F

$\mathcal{M}$ "almost always" simulates a μ-good forger

- the size of Γ_{good} defined by $\mathcal{M}$ follows a binomial distribution of parameters $(|G|, \mu)$
  $\Rightarrow$ by a Chernoff bound, $|\Gamma_{\mathrm{good}}| \simeq \mu|\mathbb{G}|$ with overwhelming probability
- in that case, the success probability of the simulated forger satisfies:

$$\varepsilon_F = 1 - (1 - \mu)^{q_h}$$

- by setting μ appropriately, $\mathcal{M}$ can simulate a forger achieving the required success probability ε_F

$\mathcal{M}$ "almost always" simulates a μ-good forger

- the size of Γ_{good} defined by $\mathcal{M}$ follows a binomial distribution of parameters $(|G|, \mu)$
  $\Rightarrow$ by a Chernoff bound, $|\Gamma_{\text{good}}| \simeq \mu|\mathbb{G}|$ with overwhelming probability
- in that case, the success probability of the simulated forger satisfies:

$$\varepsilon_F = 1 - (1 - \mu)^{q_h}$$

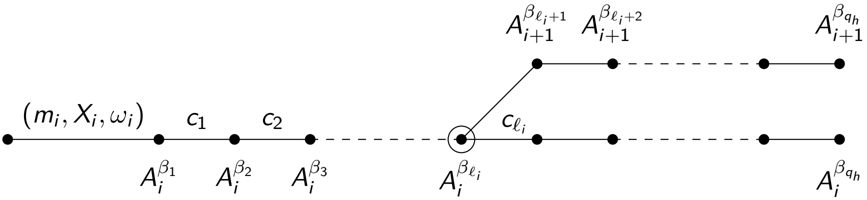
- by setting μ appropriately, $\mathcal{M}$ can simulate a forger achieving the required success probability ε_F

Probability of event Bad

- event Bad happens only if some execution forks from a previous one at the forgery point, and the new answer c' is such that $Z' = A_i^{\beta_{\ell_i}} X_i^{c'}$ is fresh and is put in Γ_{good} $\Rightarrow$ probability less than μ for each execution
- probability of Bad:

$$\Pr[\mathsf{Bad}] \leq n\mu \leq \frac{n}{g(\varepsilon_F)q_h}$$

- hence to have $\Pr[\mathsf{Bad}] \simeq 1$ one must have $n \simeq g(\varepsilon_F)q_h$ and so $\boxed{\rho_R/\rho_F \simeq f(\varepsilon_F)q_h}$

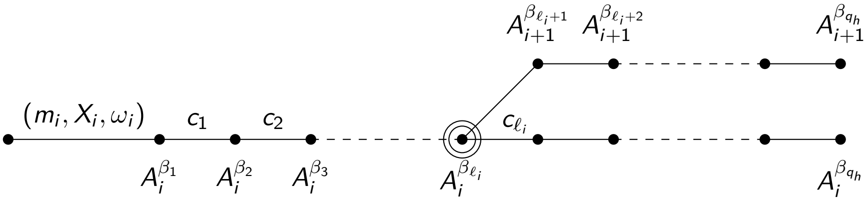


Probability of event Bad

- event Bad happens only if some execution forks from a previous one at the forgery point, and the new answer c' is such that $Z' = A_i^{\beta_{\ell_i}} X_i^{c'}$ is fresh and is put in Γ_{good} $\Rightarrow$ probability less than μ for each execution
- probability of Bad:

$$\Pr[\text{Bad}] \leq n\mu \leq \frac{n}{g(\varepsilon_F)q_h}$$

- hence to have $\Pr[\text{Bad}] \simeq 1$ one must have $n \simeq g(\varepsilon_F)q_h$ and so $\boxed{\rho_R/\rho_F \simeq f(\varepsilon_F)q_h}$

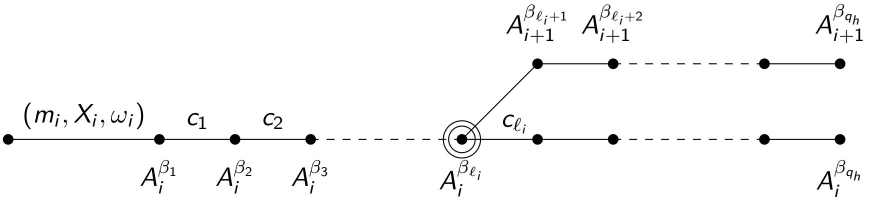


Probability of event Bad

- event Bad happens only if some execution forks from a previous one at the forgery point, and the new answer c' is such that $Z' = A_i^{\beta_{\ell_i}} X_i^{c'}$ is fresh and is put in Γ_{good} $\Rightarrow$ probability less than μ for each execution
- probability of Bad:

$$\Pr[\text{Bad}] \leq n\mu \leq \frac{n}{g(\varepsilon_F)q_h}$$

- hence to have $\Pr[\text{Bad}] \simeq 1$ one must have $n \simeq g(\varepsilon_F)q_h$ and so $\boxed{\rho_R/\rho_F \simeq f(\varepsilon_F)q_h}$

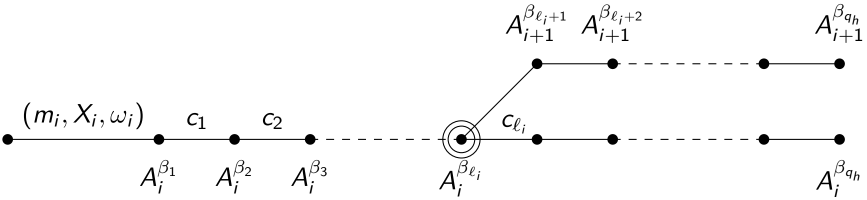


Probability of event Bad

- event $\texttt{Bad}$ happens only if some execution forks from a previous one at the forgery point, and the new answer c' is such that $Z' = A_i^{\beta_{\ell_i}} X_i^{c'}$ is fresh and is put in Γ_{good} $\Rightarrow$ probability less than μ for each execution
- probability of $\texttt{Bad}$:

$$\Pr[\texttt{Bad}] \leq n\mu \leq \frac{n}{g(\varepsilon_F) q_h}$$

- hence to have $\Pr[\texttt{Bad}] \simeq 1$ one must have $n \simeq g(\varepsilon_F) q_h$ and so $\boxed{\rho_R/\rho_F \simeq f(\varepsilon_F) q_h}$

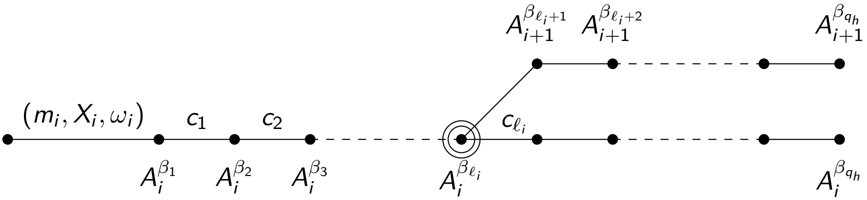


Probability of event Bad

- event Bad happens only if some execution forks from a previous one at the forgery point, and the new answer c' is such that $Z' = A_i^{\beta_{\ell_i}} X_i^{c'}$ is fresh and is put in Γ_{good} $\Rightarrow$ probability less than μ for each execution
- probability of Bad:

$$\Pr[\mathtt{Bad}] \leq n\mu \leq \frac{n}{g(\varepsilon_F)q_h}$$

- hence to have $\Pr[\mathtt{Bad}] \simeq 1$ one must have $n \simeq g(\varepsilon_F)q_h$ and so $\boxed{\rho_R/\rho_F \simeq f(\varepsilon_F)q_h}$



Expected-time and queries forgers

- considering forgers whose expected number of queries is upper bounded by q_h makes the analysis much easier
- the meta-reduction now simulates a forger which makes an a priori unbounded number of RO queries $\boldsymbol{H}(m, A_i) = c_i$ until there is a query such that $A_i X^{c_i} \in \Gamma_{\mathrm{good}}$
- if $|\Gamma_{\mathrm{good}}| = \mu|\mathbb{G}|$, the forgery index ℓ has a geometric distribution of parameter μ
- it follows that $\mathbb{E}(\#\mathrm{RO\ queries}) = 1/\mu$
  $\Rightarrow$ one can simply set $\mu = 1/q_h$
- this shows that any algebraic reduction must lose a factor q_h independently of ε_F

Expected-time and queries forgers

- considering forgers whose expected number of queries is upper bounded by q_h makes the analysis much easier
- the meta-reduction now simulates a forger which makes an a priori unbounded number of RO queries $\boldsymbol{H}(m, A_i) = c_i$ until there is a query such that $A_i X^{c_i} \in \Gamma_{\mathrm{good}}$
- if $|\Gamma_{\mathrm{good}}| = \mu|\mathbb{G}|$, the forgery index ℓ has a geometric distribution of parameter μ
- it follows that $\mathbb{E}(\#\text{RO queries}) = 1/\mu$
  $\Rightarrow$ one can simply set $\mu = 1/q_h$
- this shows that any algebraic reduction must lose a factor q_h independently of ε_F

Expected-time and queries forgers

- considering forgers whose expected number of queries is upper bounded by q_h makes the analysis much easier
- the meta-reduction now simulates a forger which makes an a priori unbounded number of RO queries $\boldsymbol{H}(m, A_i) = c_i$ until there is a query such that $A_i X^{c_i} \in \Gamma_{\mathrm{good}}$
- if $|\Gamma_{\mathrm{good}}| = \mu|\mathbb{G}|$, the forgery index ℓ has a geometric distribution of parameter μ
- it follows that $\mathbb{E}(\#\text{RO queries}) = 1/\mu$
  $\Rightarrow$ one can simply set $\mu = 1/q_h$
- this shows that any algebraic reduction must lose a factor q_h independently of ε_F

Expected-time and queries forgers

- considering forgers whose expected number of queries is upper bounded by q_h makes the analysis much easier
- the meta-reduction now simulates a forger which makes an a priori unbounded number of RO queries $\boldsymbol{H}(m, A_i) = c_i$ until there is a query such that $A_i X^{c_i} \in \Gamma_{\text{good}}$
- if $|\Gamma_{\text{good}}| = \mu|\mathbb{G}|$, the forgery index ℓ has a geometric distribution of parameter μ
- it follows that $\mathbb{E}(\#\text{RO queries}) = 1/\mu$
  $\Rightarrow$ one can simply set $\mu = 1/q_h$
- this shows that any algebraic reduction must lose a factor q_h independently of ε_F

Expected-time and queries forgers

- considering forgers whose expected number of queries is upper bounded by q_h makes the analysis much easier
- the meta-reduction now simulates a forger which makes an a priori unbounded number of RO queries $\boldsymbol{H}(m, A_i) = c_i$ until there is a query such that $A_i X^{c_i} \in \Gamma_{\text{good}}$
- if $|\Gamma_{\text{good}}| = \mu|\mathbb{G}|$, the forgery index ℓ has a geometric distribution of parameter μ
- it follows that $\mathbb{E}(\#\text{RO queries}) = 1/\mu$
  $\Rightarrow$ one can simply set $\mu = 1/q_h$
- this shows that any algebraic reduction must lose a factor q_h independently of ε_F

Extensions

The result can be extended in three ways:

- excluding tight reductions from the OMDL problem to forging Schnorr signatures (under the OMDL assumption)
- extension to generalized Schnorr signatures built from any one-way group homomorphism (Guillou-Quisquater, Okamoto. . .):
  $\Rightarrow$ any reduction from the inversion problem for the group homomorphism must lose a factor q_h, assuming the One More Inversion problem is hard
- extension to variants of Schnorr signatures, e.g. Modified ElGamal of [PS00]

Extensions

The result can be extended in three ways:

- excluding tight reductions from the OMDL problem to forging Schnorr signatures (under the OMDL assumption)
- extension to generalized Schnorr signatures built from any one-way group homomorphism (Guillou-Quisquater, Okamoto...):
  $\Rightarrow$ any reduction from the inversion problem for the group homomorphism must lose a factor q_h, assuming the One More Inversion problem is hard
- extension to variants of Schnorr signatures, e.g. Modified ElGamal of [PS00]

Extensions

The result can be extended in three ways:

- excluding tight reductions from the OMDL problem to forging Schnorr signatures (under the OMDL assumption)
- extension to generalized Schnorr signatures built from any one-way group homomorphism (Guillou-Quisquater, Okamoto...):
  $\Rightarrow$ any reduction from the inversion problem for the group homomorphism must lose a factor q_h, assuming the One More Inversion problem is hard
- extension to variants of Schnorr signatures, e.g. Modified ElGamal of [PS00]

Extensions

The result can be extended in three ways:

- excluding tight reductions from the OMDL problem to forging Schnorr signatures (under the OMDL assumption)
- extension to generalized Schnorr signatures built from any one-way group homomorphism (Guillou-Quisquater, Okamoto...):
  $\Rightarrow$ any reduction from the inversion problem for the group homomorphism must lose a factor q_h, assuming the One More Inversion problem is hard
- extension to variants of Schnorr signatures, e.g. Modified ElGamal of [PS00]

Conclusion

Bottomline

The Forking Lemma is optimal (for black-box, algebraic reductions).

- interpretation of the result: points out the limitations of black-box reduction techniques rather than a real hardness gap
- open problems:
  - what about arbitrary reductions (not nec. algebraic)?
  - what about non black-box reductions?
  - what about reductions to other problems?
  - build an efficient signature scheme with a tight reduction to the DL problem (even in the ROM this seems difficult)

Conclusion

Bottomline

The Forking Lemma is optimal (for black-box, algebraic reductions).

- interpretation of the result: points out the limitations of black-box reduction techniques rather than a real hardness gap
- open problems:
  - what about arbitrary reductions (not nec. algebraic)?
  - what about non black-box reductions?
  - what about reductions to other problems?
  - build an efficient signature scheme with a tight reduction to the DL problem (even in the ROM this seems difficult)

Conclusion

Bottomline

The Forking Lemma is optimal (for black-box, algebraic reductions).

- interpretation of the result: points out the limitations of black-box reduction techniques rather than a real hardness gap
- open problems:
  - what about arbitrary reductions (not nec. algebraic)?
  - what about non black-box reductions?
  - what about reductions to other problems?
  - build an efficient signature scheme with a tight reduction to the DL problem (even in the ROM this seems difficult)

Conclusion

Bottomline

The Forking Lemma is optimal (for black-box, algebraic reductions).

- interpretation of the result: points out the limitations of black-box reduction techniques rather than a real hardness gap
- open problems:
  - what about arbitrary reductions (not nec. algebraic)?
  - what about non black-box reductions?
  - what about reductions to other problems?
  - build an efficient signature scheme with a tight reduction to the DL problem (even in the ROM this seems difficult)

Conclusion

Bottomline

The Forking Lemma is optimal (for black-box, algebraic reductions).

- interpretation of the result: points out the limitations of black-box reduction techniques rather than a real hardness gap
- open problems:
  - what about arbitrary reductions (not nec. algebraic)?
  - what about non black-box reductions?
  - what about reductions to other problems?
  - build an efficient signature scheme with a tight reduction to the DL problem (even in the ROM this seems difficult)

Conclusion

Bottomline

The Forking Lemma is optimal (for black-box, algebraic reductions).

- interpretation of the result: points out the limitations of black-box reduction techniques rather than a real hardness gap
- open problems:
  - what about arbitrary reductions (not nec. algebraic)?
  - what about non black-box reductions?
  - what about reductions to other problems?
  - build an efficient signature scheme with a tight reduction to the DL problem (even in the ROM this seems difficult)

Conclusion

Bottomline

The Forking Lemma is optimal (for black-box, algebraic reductions).

- interpretation of the result: points out the limitations of black-box reduction techniques rather than a real hardness gap
- open problems:
  - what about arbitrary reductions (not nec. algebraic)?
  - what about non black-box reductions?
  - what about reductions to other problems?
  - build an efficient signature scheme with a tight reduction to the DL problem (even in the ROM this seems difficult)

The end. . .

Thanks for your attention!

Comments or questions?