Tight adaptive reprogramming in the QROM

Alex B. Grilo[1] Kathrin Hövelmanns[2] Andreas Hülsing[2] Christian Majenz[3]

[1]Sorbonne Université, France

[2]TU Eindhoven, The Netherlands [3]TU of Denmark, Denmark

Asiacrypt
December 08, 2021

The Quantum Random Oracle Model (QROM)

ROM:

- Efficient constructions & simple proofs

Quantum ROM (QROM):

- Generalisation: Model RO as quantum-accessible
- QROM proof $\rightarrow$ security against quantum attacks, run in classical networks
- Challenges: quantum access $\rightarrow$
  - Useful ROM properties? (programmability, preimage awareness)
  - Tightness

The Quantum Random Oracle Model (QROM)

ROM:

- Efficient constructions & simple proofs

Quantum ROM (QROM):

- Generalisation: Model RO as quantum-accessible
- QROM proof $\rightarrow$ security against quantum attacks, run in classical networks
- Challenges: quantum access $\rightarrow$
  - Useful ROM properties? (programmability, preimage awareness)
  - Tightness

Adaptive reprogramming in the (q)ROM

ROM:

- $\mathrm{RO}(x)$ not yet queried $\rightarrow$ choose $\mathrm{RO}(x)$ on the fly
- Make games simulatable without secret knowledge (e.g., sk)

QROM:

- Superposition queries $\rightarrow$ Any query might contain x
- Checking whether queries contain x might disturb the attacker

Can we adaptively reprogram in the QROM?

This talk

1. A motivational use case
2. Our results + applications
3. Proof sketch
4. Tightness of our bound: A matching attack

Motivating example:
Use case in the ROM

Fiat-Shamir signatures

Signature scheme $\mathsf{FS}[\mathsf{ID}, \mathsf{H}]$:

Built from identification scheme $\mathsf{ID} = (\mathsf{KG}, \mathsf{Comm}, \mathsf{Resp}, \mathsf{Ver}_{\mathsf{ID}})$

- Signing m:
  - $(com, st) \leftarrow \mathsf{Comm}(sk)$
  - $resp \leftarrow \mathsf{Resp}(sk, com, \mathsf{H}(com, m), st)$
  - Signature $\sigma := (com, resp)$
- Verification of $(m, \sigma = (com, resp))$:
  - use $\mathsf{Ver}_{\mathsf{ID}}$ to verify $(com, \mathsf{H}(com, m), resp)$

Proof step: ID HVZK $\overset{\text{ROM}}{\Rightarrow}$ Sign simulatable without sk

HVZK: ID transcripts $(com, chal, resp)$ simulatable by $\mathsf{Sim}(pk)$

Fiat-Shamir signatures: Simulating the Sign oracle

Goal: ID HVZK $\overset{\text{ROM}}{\Rightarrow}$ Sign simulatable without sk

Idea: Simulate $\sigma = (com, resp)$, using the HVZK simulator:

- $(com, chal, resp) \leftarrow \mathrm{Sim}(pk)$
- Signature $\sigma := (com, resp)$
- Program $\mathrm{H}(com, m) := chal$

Works unless $\mathrm{H}(com, m)$ was already queried

$\Pr \leq \#$ queries to H $\cdot \Pr[com]$ per signature

Distinguishing advantage $\leq \mathrm{HVZK}(q_S) + q_S \cdot q_H \cdot \max_{com} \Pr[com]$.

$q_H := \#$ queries to H, $q_S := \#$ queries to Sign

Fiat-Shamir signatures: Simulating the Sign oracle

Goal: ID HVZK $\overset{\text{ROM}}{\Rightarrow}$ Sign simulatable without *sk*

Idea: Simulate $\sigma = (com, resp)$, using the HVZK simulator:

- $(com, chal, resp) \leftarrow \text{Sim}(pk)$
- Signature $\sigma := (com, resp)$
- Program $\text{H}(com, m) := chal$

Works unless $\text{H}(com, m)$ was already queried

$\Pr \leq \#$ queries to H $\cdot \Pr[com]$ per signature

Distinguishing advantage $\leq \text{HVZK}(q_S) + q_S \cdot q_H \cdot \max_{com} \Pr[com]$.

$q_H := \#$ queries to H, $q_S := \#$ queries to Sign

Takeaway from the application example

We used:

- Reprogramming triggered by queries to oracle Sign
- Unlikely to query $\mathsf{H}(com, m)$ before $\mathsf{Sign}(m)$

QROM:

- What does 'H(com, m) was queried before Sign (m)' mean?
- Checkable without disturbing the attacker?

Our results

'Reprogramming toolbox': Simplest case

Distinguishing task: Access to H_0 vs adaptively reprogrammed H_1

- Start A with access to $H_0 : X_1 \times X_2 \to Y$
- A picks $x_1 \in X_1$, game uniformly picks x_2 from X_2
- Define H_1 as H_0, reprogrammed on (x_1, x_2):
  - $H_1 := H_0$ anywhere but on (x_1, x_2)
  - $H_1(x_1, x_2) := y$ for uniform $y \in Y$
- Continue A with input x_2 and access to either H_0 or H_1

Classically: Distinguishing advantage $\leq \frac{\#\text{ oracle queries}}{|X_2|}$.

Theorem

Distinguishing advantage $\leq 1.5 \cdot \sqrt{\frac{\#\text{ oracle queries}}{|X_2|}}$.

This bound is tight. (Last section of the talk)

'Reprogramming toolbox': General case

Theorem

Distinguishing advantage $\leq 1.5 \cdot \sqrt{\frac{\#\text{ queries}}{|X_2|}}$.

'Reprogramming toolbox': General case

Generalisations:

- reprogramming R many times

Theorem

Distinguishing advantage $\leq 1.5 \cdot \sqrt{\frac{\#\text{ queries}}{|X_2|}}$.

'Reprogramming toolbox': General case

Generalisations:

- reprogramming R many times

Theorem

Distinguishing advantage $\leq 1.5 \cdot R \cdot \sqrt{\frac{\#\text{ queries}}{|X_2|}}$.

'Reprogramming toolbox': General case

Generalisations:

- reprogramming R many times
- positions picked according to distributions $D_1, \cdots, D_R$, completely chosen by A

Theorem

Distinguishing advantage $\leq 1.5 \cdot R \cdot \sqrt{\frac{\#\text{ queries}}{|X_2|}}$.

'Reprogramming toolbox': General case

Generalisations:

- reprogramming R many times
- positions picked according to distributions $D_1, \cdots, D_R$, completely chosen by A

Theorem

Distinguishing advantage $\leq 1.5 \cdot R \cdot \sqrt{\# \text{ queries} \cdot \max_{x,r} \Pr[x \leftarrow D_r]}$.

'Reprogramming toolbox': General case

Generalisations:

- reprogramming R many times
- positions picked according to distributions $D_1, \cdots, D_R$, completely chosen by A
- $D_1, \cdots, D_R$ generate additional information x' that is also given to A

Theorem

Distinguishing advantage $\leq 1.5 \cdot R \cdot \sqrt{\#\text{ queries} \cdot \max_{x,r} \Pr[x \leftarrow D_r]}$.

Comparison with previous results

Reference	Bound for $R = 1$	arbitrary distribut.s	side information
Unr14a, ES15, HRS16	$\frac{2q}{\sqrt{\lvert X_2 \rvert}}$	no	no
This work	$1.5 \cdot \sqrt{q \cdot \max_{x,r} \Pr_{D_r}[x]}$	yes	yes

$q := \#$ queries to H

Reprogramming R many times: Multiply by R

Application to FS: Simulating the Sign oracle

Goal: Use same simulation as in ROM

Application to FS: Simulating the Sign oracle

Goal: Use same simulation as in ROM

Honest signatures $\sigma = (\mathit{com}, \mathit{resp})$:

$$(\mathit{com}, \mathit{st}) \leftarrow \mathrm{Comm}(\mathit{sk})$$
$$\mathit{chal} := \mathrm{H}(\mathit{com}, m)$$
$$\mathit{resp} \leftarrow \mathrm{Resp}(\mathit{sk}, \mathit{com}, \mathit{chal}, \mathit{st})$$

Application to FS: Simulating the Sign oracle

Goal: Use same simulation as in ROM

Honest signatures $\sigma = (\mathit{com}, \mathit{resp})$: $\rightarrow$ Intermediate simulation:

$(\mathit{com}, \mathit{st}) \leftarrow \mathrm{Comm}(\mathit{sk})$

$\mathit{chal} := \mathrm{H}(\mathit{com}, m)$

$\mathit{resp} \leftarrow \mathrm{Resp}(\mathit{sk}, \mathit{com}, \mathit{chal}, \mathit{st})$

Use uniform *chal* instead

Program $\mathrm{H}(\mathit{com}, m) := \mathit{chal}$

Application to FS: Simulating the Sign oracle

Goal: Use same simulation as in ROM

Honest signatures $\sigma = (com, resp)$: $\rightarrow$ Intermediate simulation:

$(com, st) \leftarrow \mathrm{Comm}(sk)$

$chal := \mathrm{H}(com, m)$

$resp \leftarrow \mathrm{Resp}(sk, com, chal, st)$

Use uniform *chal* instead

Program $\mathrm{H}(com, m) := chal$

Works due to our theorem

Dist. advantage $\leq 1.5 q_S \cdot \sqrt{(q_S + q_H + 1) \cdot \max_{com} \Pr_{\mathrm{Comm}}[com]}$

Application to FS: Simulating the Sign oracle

Goal: Use same simulation as in ROM

Intermediate simulation:

Use uniform *chal* instead

Program $\mathsf{H}(com, m) := chal$

Application to FS: Simulating the Sign oracle

Goal: Use same simulation as in ROM

Intermediate simulation:	$\rightarrow$ Desired simulation:
Use uniform *chal* instead	$(com, chal, resp) \leftarrow \mathrm{Sim}(pk)$
Program $\mathrm{H}(com, m) := chal$	Program $\mathrm{H}(com, m) := chal$

Application to FS: Simulating the Sign oracle

Goal: Use same simulation as in ROM

Intermediate simulation:	$\rightarrow$ Desired simulation:
Use uniform *chal* instead	$(com, chal, resp) \leftarrow \mathsf{Sim}(pk)$
Program $\mathsf{H}(com, m) := chal$	Program $\mathsf{H}(com, m) := chal$

Intermediate $\rightarrow$ desired: Justifiable via HVZK

Summary of our applications

Fiat-Shamir signatures:

- Conceptually simple proof that is tighter than previous ones

XMSS:

- Tighter proof due to tighter proof for hash-and-sign

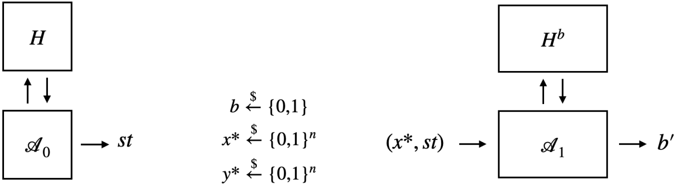
Hedged Fiat-Shamir:

- Resistance against fault injections and nonce attacks

Proof technique

ROM proof sketch

For simplicity: trivial X_1 and $X_2 = X = Y = \{0,1\}^n$



Theorem

Distinguishing advantage $\leq \frac{\#\text{ queries}}{2^n}$.

Intuition: 2 ways for the adversary to win:

1. learn $H(x^*)$ in learning phase
2. guess b.

$$\Rightarrow \text{Advantage} \leq \mathbb{E}_{x*} \Pr[x^* \text{ has been queried in learning phase}]$$
$$\leq \frac{\#\text{ queries}}{2^n}$$

What is a superposition oracle?

Essentially, it's a "quantum function table" of a random function

for simplicity: n bit to n bit random function

	random oracle function table	superposition oracle
entire object	2^n random variables	2^n quantum registers
unqueried entry	independent uniform RV	uniform superposition state

Proof sketch

Recall ROM proof sketch

$$\text{Advantage} \leq \mathbb{E}_{x*} \Pr[x^* \text{ has been queried in learning phase}]$$
$$\leq \frac{\# \text{ queries}}{2^n}$$

Works because

	random oracle function table	superposition oracle
entire object	2^n random variables	2^n quantum registers
unqueried entry	uniform RV	uniform superposition state

In QROM:

- use superposition oracle
- prepare $H(x^*)$ register in fresh uniform superposition instead of $y^* \leftarrow \{0,1\}^n$
- use "$\mathbb{E}_{x*} \Pr[H(x^*)$ register is not in uniform superposition]" instead of $\mathbb{E}_{x*} \Pr[x^*$ has been queried in learning phase]

A matching attack

Classical attack

Theorem

Distinguishing advantage $\leq \frac{q}{2^n}$.

$q = \#$ of queries

Is this tight?

Simple attack: Query q different values of x, store results and hope that the reprogrammed input is among the queried ones.

q queries, but memory cost $= O(q)$!

Better: Query q different values of x, store **XOR of** results and hope that the reprogrammed input is among the queried ones.

$2q$ queries, memory cost $= O(1)$

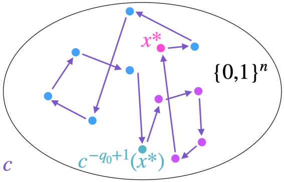
Quantum attack

Better: Query q different values of x, store **XOR of** results and hope that the reprogrammed input is among the queried ones.

Succeeds if $x^* \in S$ (set of queried inputs), $\Pr[x^* \in S] = q \cdot 2^{-n}$.

Idea: query superposition of S's, sucess probability should grow with **amplitude** of $x^* \in S$

$\Rightarrow$ start with uniform superposition, repeatedly query and apply some fixed cyclic permutation. After reprogramming, undo.



Theorem

Distinguishing advantage $O\left(\sqrt{\frac{\#\text{ oracle queries}}{2^n}}\right)$ is achievable.

Thanks for listening!

Slides:
hoevelmanns.net

handles:
Alex: @abgrilo
Kathrin: @quantum_bat
Andreas: @cr_yp_to
Christian: @cmajenz