Rewinding

Advanced Topics in ~~Cybersecurity~~Cryptography (7CCSMATC)

Martin R. Albrecht

Introduction

Main Reference

Arno Mittelbach and Marc Fischlin. **Chapter 10: Random Oracle Schemes in Practice**. In: *The Theory of Hash Functions and Random Oracles - An Approach to Modern Cryptography*. Information Security and Cryptography. Springer, 2021. ISBN: 978-3-030-63286-1. DOI: 10.1007/978-3-030-63287-8. URL: https://doi.org/10.1007/978-3-030-63287-8

Recap: Schnorr Signatures

Let $H : \mathbb{G} \times \{0,1\}^* \to \mathbb{Z}_p$ be a hash function

Claus-Peter Schnorr. **Efficient Identification and Signatures for Smart Cards**. In: *CRYPTO'89*. Ed. by Gilles Brassard. Vol. 435. LNCS. Springer, New York, Aug. 1990, pp. 239–252. DOI: `10.1007/0-387-34805-0_22`

Gen

$\mathsf{sk} := x \leftarrow\!\!\$\ \mathbb{Z}_p; \quad \mathsf{vk} := X \leftarrow G^x$

Sign(sk, m)

1. $y \leftarrow\!\!\$\ \mathbb{Z}_p$ and set $Y \leftarrow G^y$
2. $c \leftarrow H(Y, m)$
3. $z \leftarrow y - c \cdot x$

$\sigma := (Y, z)$

Verify(vk, m)

1. $c \leftarrow H(Y, m)$
2. $Y \stackrel{?}{=} G^z \cdot X^c = G^z \cdot G^{c \cdot x} = G^{y - c\cdot x + c \cdot x}$

Recap: Proof

The proof of this (ubiquitous) construction relies on two proof techniques we have yet to cover:

- *the Random Oracle Model*
- *Rewinding*

Recap: Proof

The proof of this (ubiquitous) construction relies on two proof techniques we have yet to cover:

- *the Random Oracle Model* ✓
- *Rewinding* ***YOU ARE HERE***

Schnorr Identification Scheme

Schnorr Identification Scheme

Alice		**Bob**
knows: x s.t. $G^x \equiv X$		knows: X
$y \leftarrow\!\!\$\ \mathbb{Z}_p, Y \leftarrow G^y$	$\xrightarrow{\quad Y \quad}$	
	$\xleftarrow{\quad c \quad}$	$c \leftarrow\!\!\$\ \mathbb{Z}_p$
$z \leftarrow y - c \cdot x$	$\xrightarrow{\quad z \quad}$	
		assert $Y \equiv G^z \cdot X^c$

Jargon: Transcript

Definition (Transcript)

A **transcript** of a Schnorr protocol execution consists of the values (X, Y, c, z). It is an **accepting** transcript if $Y \equiv G^z \cdot X^c$, i.e. if Bob would accept.

Programme of Work

1. We want show that if a Alice convinces Bob she must “know” x.
    - What does it even mean for a program to “know” some value? It could be encoded in any which way in its code.

Programme of Work

1. We want show that if a Alice convinces Bob she must “know” x.
    - What does it even mean for a program to “know” some value? It could be encoded in any which way in its code.
    - We will prove that Alice must “know” x by extracting it from the messages she sends.

Programme of Work

1. We want show that if a Alice convinces Bob she must “know” x.
    - What does it even mean for a program to “know” some value? It could be encoded in any which way in its code.
    - We will prove that Alice must “know” x by extracting it from the messages she sends.
2. We want to show, at least, that anyone observing the messages being sent **cannot** learn anything about secret value x except that Alice “knows” it.

Contradiction!

We want our cake (extract x) and eat it (keep x hidden)!

Programme of Work

1. We want show that if a Alice convinces Bob she must "know" x.
    - What does it even mean for a program to "know" some value? It could be encoded in any which way in its code.
    - We will prove that Alice must "know" x by extracting it from the messages she sends.
2. We want to show, at least, that anyone observing the messages being sent **cannot** learn anything about secret value x except that Alice "knows" it.

Contradiction!

We want our cake (extract x) and eat it (keep x hidden)!

The magic that makes this work is **rewinding**!

Proof Idea 1: Knowledge Soundness

Assume Alice can answer for at least **two** different challenges c, c'.

Lemma (Special Soundness)

There exists an efficient algorithm that computes x from X, given any two accepting transcripts (X, Y, c, z) and (X, Y, c', z') where $c' \neq c$.

1. $Y \equiv G^z \cdot X^c$ and $Y \equiv G^{z'} \cdot X^{c'}$
2. $G^z \cdot X^c \equiv G^{z'} \cdot X^{c'}$
3. $G^{z-z'} \equiv X^{c'-c}$
4. $G^{(z-z')/(c'-c)} \equiv X$

Proof Idea 2: Zero-Knowledge

This argument critically relies on Alice picking (Y, y) before knowing c or c'

Alice samples $z \leftarrow\$ \mathbb{Z}_p$ and outputs it together with $Y := G^z \cdot X^{-c}$

Bob verifies $Y \equiv G^z \cdot X^c$

We can leverage this to prove that the signature scheme does not leak x!

Texas sharpshooter fallacy. The Texas sharpshooter fires randomly at a barn door and then paints the targets around the bullet holes, creating the false impression of being an excellent marksman.

Zero-Knowledge

Schnorr(x)	Simulated(x)
$X \leftarrow G^x$	$X \leftarrow G^x$
$y \leftarrow\!\!\$\ \mathbb{Z}_p$	$c \leftarrow\!\!\$\ \mathbb{Z}_p$
$Y \leftarrow G^y$	$z \leftarrow\!\!\$\ \mathbb{Z}_p$
$c \leftarrow\!\!\$\ \mathbb{Z}_p$	$Y \leftarrow G^z \cdot X^{-c}$
$z \leftarrow y - c \cdot x \bmod p$	
return (X, Y, c, z)	**return** (X, Y, c, z)

We want to show that these two games are indistinguishable.

Proof of Zero-Knowledge Property

$\mathsf{Game}_0(x)$	$\mathsf{Game}_1(x)$	$\mathsf{Game}_2(x)$	$\mathsf{Game}_3(x)$
$X \leftarrow G^x$	$X \leftarrow G^x$	$X \leftarrow G^x$	$X \leftarrow G^x$
$y \leftarrow\!\!\$\ \mathbb{Z}_p$	$c \leftarrow\!\!\$\ \mathbb{Z}_p$	$c \leftarrow\!\!\$\ \mathbb{Z}_p$	$c \leftarrow\!\!\$\ \mathbb{Z}_p$
$Y \leftarrow G^y$	$y \leftarrow\!\!\$\ \mathbb{Z}_p$	$z \leftarrow\!\!\$\ \mathbb{Z}_p$	$z \leftarrow\!\!\$\ \mathbb{Z}_p$
$c \leftarrow\!\!\$\ \mathbb{Z}_p$	$z \leftarrow y - c \cdot x$	$y \leftarrow z + c \cdot x$	
$z \leftarrow y - c \cdot x$	$Y \leftarrow G^y$	$Y \leftarrow G^y$	$Y \leftarrow G^z \cdot X^c$
return (X, Y, c, z)	**return** (X, Y, c, z)	**return** (X, Y, c, z)	**return** (X, Y, c, z)

Caveat

This proof only works for adversaries observing a transcript that was correctly computed. In particular, we assume that c is chosen uniformly at random.

This property is called "Honest-Verified Zero-Knowledge" (HVZK).

- It seems quite limiting to assume the verifier (here the adversary) behaves honestly
- It turns out to be quite useful, see below.

Soundness

Soundness

- We can extract x if we convince Alice to respond correctly to two different challenges c and c' for the same Y.
- How do we convince Alice? We do not need to, because we remember that Alice is a program and we control its execution environment.
    1. Run Alice in a VM.
    2. Wait until Alice has output Y.
    3. Take a snapshot of the VM.
    4. Continue to run Alice twice from the same snapshot, sending different c and c'.

This is what we call **rewinding**.

A Note: Quantum Adversaries

No-Cloning Theorem

Cannot **in general** rewind Quantum Alice.

See quantum lecture.

Success?

- Say, Alice is a prover but only answers a fraction ε of all challenge queries correctly.
- Can we still use our strategy to show that Alice cannot be a cheating prover and must "know" x?
- Put differently, with what probability can we extract x?
- We know:
  1. Alice responds successfully to challenges over the randomness of her choice Y and the randomness of the challenges c.
  2. If Alice responds successfully to two challenges c, c' for the same Y then we can extract x.
- What is the probability of (2) knowing that for (1) it is ε?

Two Variants

"Heavy-Row Argument"

We do not have to rewind too often until we can extract x.

Ivan Damgård. On Σ-protocols. In: *Lecture Notes, University of Aarhus, Department for Computer Science* 84 (2002). `https://www.cs.au.dk/~ivan/Sigma.pdf`

"Forking Lemma"

If we rewind once we extract x with some decent probability.

David Pointcheval and Jacques Stern. Security Proofs for Signature Schemes. In: *EUROCRYPT'96.* Ed. by Ueli M. Maurer. Vol. 1070. LNCS. Springer, Berlin, Heidelberg, May 1996, pp. 387–398. DOI: `10.1007/3-540-68339-9_33`

These arguments are ubiquitous in security proofs in cryptography.

The Kernel of the Argument I

- Consider a massive matrix with $0, 1$ entries. The rows are indexed by all possible choices of Y and the columns are index by all possible choices of c.
- In our case the matrix can be forced to have dimensions $p \times p$.
- Something like the matrix on right:
- We have $\mathbf{H}_{Y_i, c_j} = 1$ when Alice outputs an accepting transcript and zero other.
- We cannot write $\mathbf{H}$ down but, given access to Alice, we can probe entries of $\mathbf{H}$.
- By rewinding, we can repeatedly probe a single row of $\mathbf{H}$.

$$\mathbf{H} := \begin{array}{c c} & \begin{array}{ccccc} c_0 & c_1 & c_2 & c_3 & \cdots \end{array} \\ \begin{array}{c} Y_0 \\ Y_1 \\ Y_2 \\ Y_3 \\ \vdots \end{array} & \begin{pmatrix} 1 & 1 & 0 & 1 & \cdots \\ 0 & 1 & 0 & 0 & \cdots \\ 0 & 0 & 1 & 0 & \cdots \\ 0 & 1 & 0 & 1 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{pmatrix} \end{array}$$

The Kernel of the Argument II

We know that the fraction of "1" entries in $\mathbf{H}$ is ε from (1), but we know nothing about their distribution, there might be lots of rows with a single "1" in them.

- We call a row **heavy** if it contains a fraction of at least $\varepsilon/2$ "1"s.
- We write $\#\mathbf{H}$ for the number of entries in $\mathbf{H}$. For an $n \times m$ matrix this is $n \cdot m$. So here $\#\mathbf{H} = p^2$.
- We write $\mathrm{hw}(\mathbf{H})$ for the number of "1"s in $\mathbf{H}$.

The Kernel of the Argument III

Split the rows of H into H_h with heavy rows and H_ℓ with the remaining rows (i.e. fewer "1"s than a fraction of $\varepsilon/2$.)

- $\mathrm{hw}(\mathsf{H}) = \varepsilon \cdot \#\mathsf{H}$
- $\mathrm{hw}(\mathsf{H}_\ell) < \varepsilon/2 \cdot \#\mathsf{H}_\ell$
- $\mathrm{hw}(\mathsf{H}_h) > \varepsilon \cdot \#\mathsf{H} - \varepsilon/2 \cdot \#\mathsf{H}_\ell \geq \varepsilon \cdot \#\mathsf{H} - \varepsilon/2 \cdot \#\mathsf{H} = \varepsilon/2 \cdot \#\mathsf{H}$

The Kernel of the Argument IV

- We run Alice until we get an accepting transcript. By $\mathrm{hw}(\mathbf{H}_h) > \varepsilon/2 \cdot \#\mathbf{H}$ we hit a heavy row with probability $> 1/2$.
- If we're unlucky, that's tough luck. Note that we cannot check if we are luck or unlucky. We have to proceed as if we are lucky.
- If we now randomly probe the same row (i.e. rewind and try a different challenge) again, we succeed with probability

$$\frac{\varepsilon/2 \cdot p - 1}{p} = \varepsilon/2 - 1/p.$$

Punchline

Overall we succeed with probability $> \frac{1}{2} \cdot (\varepsilon/2 - 1/p)$ once we found one accepting transcript.

Fiat-Shamir

Removing Interaction with Random Oracles

Alice		**Bob**
knows: x s.t. $G^x \equiv X$		knows: X
$y \leftarrow\$ \mathbb{Z}_p, Y \leftarrow G^y$		
$c \leftarrow \mathrm{H}(Y)$		$c \leftarrow \mathrm{H}(Y)$
$z \leftarrow y - c \cdot x$	$\xrightarrow{Y, z}$	
		assert $Y \equiv G^z \cdot X^c$

- We can replace Bob by a Random Oracle that we simply call ourselves
- This is why Honest-Verified Zero-Knowledge is sufficient
- We then program the RO to give two different answers in the two different runs of Alice

Amos Fiat and Adi Shamir. How to Prove Yourself: Practical Solutions to Identification and Signature Problems. In: *CRYPTO'86*. Ed. by Andrew M. Odlyzko. Vol. 263. LNCS. Springer, Berlin, Heidelberg, Aug. 1987, pp. 186–194. DOI: 10.1007/3-540-47721-7_12

Fiat-Shamir

Amos Fiat and Adi Shamir. How to Prove Yourself: Practical Solutions to Identification and Signature Problems. In: *CRYPTO'86*. Ed. by Andrew M. Odlyzko. Vol. 263. LNCS. Springer, Berlin, Heidelberg, Aug. 1987, pp. 186–194. DOI: `10.1007/3-540-47721-7_12`

Schnorr Signatures: Make c depend on m too!

Let $H : \mathbb{G} \times \{0,1\}^* \to \mathbb{Z}_p$ be a hash function

Gen

sk $:= x \leftarrow\$ \mathbb{Z}_p$; vk $:= X := G^x$

Claus-Peter Schnorr. Efficient Identification and Signatures for Smart Cards. In: *CRYPTO'89*. Ed. by Gilles Brassard. Vol. 435. LNCS. Springer, New York, Aug. 1990, pp. 239–252. DOI: 10.1007/0-387-34805-0_22

Sign(sk, m)

1. $y \leftarrow\$ \mathbb{Z}_p$ and set $Y \leftarrow G^y$
2. $c \leftarrow H(Y, m)$
3. $z \leftarrow y - c \cdot x$

$\sigma := (Y, z)$

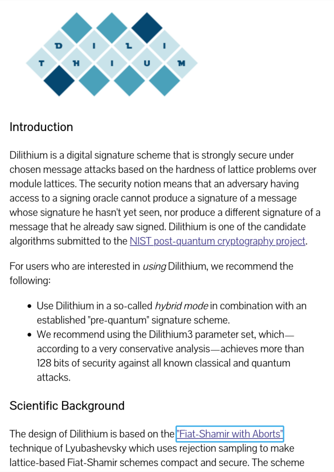
Verify(vk, m)

1. $c \leftarrow H(Y, m)$
2. $Y \stackrel{?}{=} G^z \cdot X^c = G^z \cdot G^{c \cdot x} = G^{y - c\cdot x + c\cdot x}$

If Schnorr signatures are that great, why is no one using them?

Patents (EC)DSA appears to be essentially a hack to get around Schnorr's patent (expired in 2010)

Dilithium (ML-DSA) is a lattice-based post-quantum variant of Schnorr, coming soon to a browser near you



Introduction

Dilithium is a digital signature scheme that is strongly secure under chosen message attacks based on the hardness of lattice problems over module lattices. The security notion means that an adversary having access to a signing oracle cannot produce a signature of a message whose signature he hasn't yet seen, nor produce a different signature of a message that he already saw signed. Dilithium is one of the candidate algorithms submitted to the NIST post-quantum cryptography project.

For users who are interested in *using* Dilithium, we recommend the following:

- Use Dilithium in a so-called *hybrid mode* in combination with an established "pre-quantum" signature scheme.
- We recommend using the Dilithium3 parameter set, which—according to a very conservative analysis—achieves more than 128 bits of security against all known classical and quantum attacks.

Scientific Background

The design of Dilithium is based on the "Fiat-Shamir with Aborts" technique of Lyubashevsky which uses rejection sampling to make lattice-based Fiat-Shamir schemes compact and secure. The scheme

Soundness Proof = Attack

- This security proof is also a side-channel attack: if we can make Alice sign two different messages for the same Y we can learn her signing key.
- For example, Alice's computer might not have gathered enough entropy after boot by the time she signs.

Attack = Soundness Proof I

- We can even learn the key if only a few MSBs of y_i match.
- We get:
  - $z_i := y_i - c_i \cdot x$ and thus
  - $z_0 - z_1 = y_0 - y_1 - (c_1 - c_0) \cdot x$
  - We know $c_0 + c_1$ and we know $y_0 - y_i$ is small since the MSBs match
- Similarly, if we happen to know the most significant bits of y_i we can simply subtract them and make y_i small, too.

Does this remind you of anything?

Attack = Soundness Proof I

- We can even learn the key if only a few MSBs of y_i match.
- We get:
  - $z_i := y_i - c_i \cdot x$ and thus
  - $z_0 - z_1 = y_0 - y_1 - (c_1 - c_0) \cdot x$
  - We know $c_0 + c_1$ and we know $y_0 - y_i$ is small since the MSBs match
- Similarly, if we happen to know the most significant bits of y_i we can simply subtract them and make y_i small, too.

Does this remind you of anything?

This is Learning with Errors where $n = 1$!

Attack = Soundness Proof II

This attack, in turn, takes inspiration from another security proof:

Dan Boneh and Ramarathnam Venkatesan. **Hardness of Computing the Most Significant Bits of Secret Keys in Diffie-Hellman and Related Schemes**. In: *CRYPTO'96*. Ed. by Neal Koblitz. Vol. 1109. LNCS. Springer, Berlin, Heidelberg, Aug. 1996, pp. 129–142. DOI: 10.1007/3-540-68697-5_11

State of the art in lattice attacks in this setting:

Martin R. Albrecht and Nadia Heninger. **On Bounded Distance Decoding with Predicate: Breaking the "Lattice Barrier" for the Hidden Number Problem**. In: *EUROCRYPT 2021, Part I*. ed. by Anne Canteaut and François-Xavier Standaert. Vol. 12696. LNCS. Springer, Cham, Oct. 2021, pp. 528–558. DOI: 10.1007/978-3-030-77870-5_19

Avoiding Rewinding

Rewinding, Reconsidered

- We cannot rewind **Quantum Alice** in general.
- We need rewinding to prove **knowledge soundness**, i.e. to extract x.
- What if we proved **soundness** (without the "knowledge" qualifier) directly? Can we avoid rewinding then?

Main Reference

Eu-Jin Goh, Stanislaw Jarecki, Jonathan Katz, and Nan Wang. **Efficient Signature Schemes with Tight Reductions to the Diffie-Hellman Problems**. In: *Journal of Cryptology* 20.4 (Oct. 2007), pp. 493–514. DOI: `10.1007/s00145-007-0549-3`

See

- Michel Abdalla, Pierre-Alain Fouque, Vadim Lyubashevsky, and Mehdi Tibouchi. **Tightly Secure Signatures From Lossy Identification Schemes**. In: *Journal of Cryptology* 29.3 (July 2016), pp. 597–631. DOI: `10.1007/s00145-015-9203-7`

for a post-quantum lattice-based version.

Interactive Protocol I

Given G, H, Y_0, Y_1

Alice		**Bob**
knows $G^x = Y_0$ and $H^x = Y_1$		
$y \leftarrow\!\$\ \mathbb{Z}_p, A \leftarrow G^y, B \leftarrow H^y$	$\xrightarrow{A, B}$	
	$\xleftarrow{c}$	$c \leftarrow\!\$\ \mathbb{Z}_p$
$z \leftarrow y - c \cdot x$	$\xrightarrow{z}$	
		assert $A \equiv G^z \cdot Y_0^c \wedge B \equiv G^z \cdot Y_1^c$

Completeness (Correctness)

We check that A equals $G^z \cdot Y_0^c$. Let's plug in:

$$A = G^y \stackrel{?}{\equiv} G^z \cdot Y_0^c \equiv G^{y-c\cdot x} \cdot Y_0^c \equiv G^{y-c\cdot x} \cdot (G^x)^c$$
$$\equiv G^{y-c\cdot x} \cdot G^{cx} \equiv G^{y-c\cdot x+cx} \equiv G^y$$

The argument for B and H is exactly the same.

What is going on here?

- Alice does not want to reveal x so she demonstrates to Bob that she can do consistent computations with x.
- When those check out, Bob accepts that the only way she can do that is if there **exists** some x.
    - **We have yet to prove this!**
- Alice does not prove that she “knows” x, only that $Y_0 = G^x$ and $Y_1 = H^x$ which is what allows us to avoid rewinding!

Fiat-Shamir Transform

- We can apply the Fiat-Shamir Transform as with Schnorr's scheme to make the scheme non-interactive.
- We again make the challenge c depend on m to produce a signature scheme.

The Scheme

- We'll make use of two random oracles G() and H().
- We assume there is some generator $G \in \mathbb{G}$. Pick a random $H \in \mathbb{G}$.

KeyGen $x \leftarrow\!\$\ \mathbb{Z}_p$, set $Y_0 \leftarrow G^x$, $Y_2 \leftarrow H^x$, $vk := (G, H, Y_0, Y_1)$ and $sk := x$.

Sign

1. Compute $y \leftarrow \mathsf{G}(x, m)$
2. Compute $A \leftarrow G^y$, $B \leftarrow H^y$
3. Compute $c \leftarrow \mathsf{H}(G, H, Y_0, Y_1, A, B, m)$.
4. Compute $z = y - c \cdot x \mod p$ and return $\sigma := (c, z)$.

Verify

1. Compute $A' \leftarrow G^z \cdot Y_0^c$ and $B' \leftarrow H^z \cdot Y_1^c$.
2. Accept if $c \stackrel{?}{=} \mathsf{H}(G, H, Y_0, Y_1, A', B', m)$ otherwise reject.

Proof Sketch: Security Model I

- We are playing a game with an adversary where the adversary can ask for signatures of arbitrary messages until it outputs a $(m^\star, \sigma^\star)$ that passes **Verify** but $\sigma^\star$ was never returned by the signing oracle on input message $m^\star$.
- In other words, it has output a forgery.

Proof Sketch: Security Model II

- This is known as strong existential unforgability under chosen message attacks (SUF-CMA).
- We have also seen existential unforgeability under chosen message attacks before (EUF-CMA), which is weaker by requiring m^* to have never been queried to the signing oracle.
- Here we allow it having been queried but merely insist that the signing oracle did not output $\sigma^\star$.
  - If we did not rule that out, the scheme would trivially allow for forgeries: ask for a signature from the signing oracle.

Proof Sketch: Proof Goal I

- We will show that we can use an adversary producing such a forgery into one that decide if a tuple (G, G^x, G^y, Z) is such that $Z = G^{xy}$ or Z is just some random, unrelated element.
- In other words if we have a Diffie-Hellman tuple or not.
- Since we assume that this is hard on a classical computer (this is the DDH assumption), so this implies forging signatures is hard.

Proof Sketch: Proof Goal ii

- We will do the same thing as in the FO transform: put the adversary in a box where we simulate oracles for it.
- In particular, we need to simulate the signing oracle without knowing x. We will use our random oracle H() for that
    - This is the same strategy for proving zero-knowledge as before with rewinding
- We don't really use G() in the proof except for calling it.

Proof Sketch: Proof Goal III

- We have some tuple (G, G^x, G^y, Z), for which we want to decide if $X = G^{xy}$
- We will pass this tuple to the adversary as vk of the signature scheme it is attacking.

Proof Sketch: Simulating the Signing Oracle I

When the adversary asks for a signature for the message m we check if this message was queried before.

- If yes, return the same signature σ we issued before.
- This is consistent with the real scheme where y is computed from m and thus the signature that is output will be the same when the same m is signed again.

Proof Sketch: Simulating the Signing Oracle II

- To respond to a fresh signature query, pick a random c and z and compute $A = G^z \cdot Y_0^{-c}$ and $B = H^z \cdot Y_1^{-c}$.
- We check if H() has been previously queried on $(G, H, Y_0, Y_1, A, B, m)$.
  - **If it has, we abort the simulation and concede defeat!**
- If not, we **define** c as the output of $H(G, H, Y_0, Y_1, A, B, m)$ and return (c, z).
- We're making up the answer of the Random Oracle on the spot.
  - Again, this is known as "programming" the Random Oracle.
  - We are using the fact here that we control the RO so we can choose what to return as long as it looks random.
  - The simulation trick is again: if we control the RO we can fix up the RO **after** we have picked c and z.

Proof Sketch: Implementing the Random Oracle

- When the adversary queries the random oracle on some input $(G, H, Y_0, Y_1, A, B, m)$, we first check if we had programmed it previously to some value c and return that.
- Otherwise, we pick a random value, make a note of that, and return it.

Proof Sketch: Simulation

Summary:

- We can simulate signing – using our trick of swapping the order in which we compute things by virtue of controlling the RO – successfully, unless the adversary somehow managed to query $\mathrm{H}(G, H, Y_0, Y_1, A, B, m)$ before.
- This means it has guessed y correctly to compute $A = G^y$ and $B = G^y$.
- We can show that this probaliltiy is exponentially small, but I'll avoid the details here.

Now, we only need to turn the adversary's success into our own. We do this by noting that the adversary **cannot win** if our input (G, H, Y_1, Y_2) is not a DH tuple.

Proof Sketch: Solving DDH I

Lemma

Assume $Y_1 \in \mathbb{G}$ but there is no x s.t. $G^x = Y_0$ and $H^x = Y_1$. Then for any A, B output by a cheating Alice, there is at most one value of c for which Bob will accept.

Proof.

Say $A, B \in \mathbb{G}$ are such that the prover can send correct responses z_0, z_1 for two different challenges c_0, c_1. Then

$$A = G^{z_0} \cdot Y_0^{c_0} = G^{z_1} \cdot Y_0^{c_1} \text{ and } B = H^{z_0} \cdot Y_1^{c_0} = H^{z_1} \cdot Y_1^{c_1}.$$

Noting that $c_1 \neq c_0$ we have

$$Y_0 = G^{(z_0 - z_1)/(c_1 - c_0)} \text{ and } H^{(z_0 - z_1)/(c_1 - c_0)} = Y_1$$

contrary to the assumption. $\square$

Proof Sketch: Solving DDH II

- When
$$(G, G^y, G^x, Z) = (G, G^y, G^x, G^{xy}) = (G, H, Y_0, Y_1) = (G, H, G^x, H^x)$$
then the adversary "lives" exactly in the world where it succeeds, i.e. it will output a forgery (with some probability not discussed here).
- When (G, G^y, G^x, Z) for some random =Z then the *vk* the adversary sees is not a valid *vk* for the signature scheme.
    - In particular, for all but one c there is no z that satisfies the "check equations": $A = G^z \cdot Y_0^c$ and $B = H^z \cdot Y_1^c$.
    - But c is the output of a random oracle, i.e. random.
    - The probability of hitting that one magical c for which a solution even exists is negligible.
    - In other words, whatever the adversary does it cannot win (except with very low probability).

Proof Sketch: Punchline

Thus, when the adversary wins we conclude we have a DH tuple. We have solved DDH, which we assumed cannot be done and thus have proven our scheme secure.

Fin

"Knowledge Soundness" $\neq$ "Soundness"

References I

[AFLT16] Michel Abdalla, Pierre-Alain Fouque, Vadim Lyubashevsky, and Mehdi Tibouchi. Tightly Secure Signatures From Lossy Identification Schemes. In: *Journal of Cryptology* 29.3 (July 2016), pp. 597–631. DOI: 10.1007/s00145-015-9203-7.

[AH21] Martin R. Albrecht and Nadia Heninger. On Bounded Distance Decoding with Predicate: Breaking the “Lattice Barrier” for the Hidden Number Problem. In: *EUROCRYPT 2021, Part I*. Ed. by Anne Canteaut and François-Xavier Standaert. Vol. 12696. LNCS. Springer, Cham, Oct. 2021, pp. 528–558. DOI: 10.1007/978-3-030-77870-5_19.

[BV96] Dan Boneh and Ramarathnam Venkatesan. Hardness of Computing the Most Significant Bits of Secret Keys in Diffie-Hellman and Related Schemes. In: *CRYPTO'96*. Ed. by Neal Koblitz. Vol. 1109. LNCS. Springer, Berlin, Heidelberg, Aug. 1996, pp. 129–142. DOI: 10.1007/3-540-68697-5_11.

[Dam02] Ivan Damgård. On Σ-protocols. In: *Lecture Notes, University of Aarhus, Department for Computer Science* 84 (2002). `https://www.cs.au.dk/~ivan/Sigma.pdf`.

[FS87] Amos Fiat and Adi Shamir. How to Prove Yourself: Practical Solutions to Identification and Signature Problems. In: *CRYPTO'86*. Ed. by Andrew M. Odlyzko. Vol. 263. LNCS. Springer, Berlin, Heidelberg, Aug. 1987, pp. 186–194. DOI: `10.1007/3-540-47721-7_12`.

[GJKW07] Eu-Jin Goh, Stanislaw Jarecki, Jonathan Katz, and Nan Wang. Efficient Signature Schemes with Tight Reductions to the Diffie-Hellman Problems. In: *Journal of Cryptology* 20.4 (Oct. 2007), pp. 493–514. DOI: `10.1007/s00145-007-0549-3`.

[MF21] Arno Mittelbach and Marc Fischlin. Chapter 10: Random Oracle Schemes in Practice. In: *The Theory of Hash Functions and Random Oracles - An Approach to Modern Cryptography*. Information Security and Cryptography. Springer, 2021. ISBN: 978-3-030-63286-1. DOI: 10.1007/978-3-030-63287-8. URL: https://doi.org/10.1007/978-3-030-63287-8.

[PS96] David Pointcheval and Jacques Stern. Security Proofs for Signature Schemes. In: *EUROCRYPT'96*. Ed. by Ueli M. Maurer. Vol. 1070. LNCS. Springer, Berlin, Heidelberg, May 1996, pp. 387–398. DOI: 10.1007/3-540-68339-9_33.

[Sch90] Claus-Peter Schnorr. Efficient Identification and Signatures for Smart Cards. In: *CRYPTO'89*. Ed. by Gilles Brassard. Vol. 435. LNCS. Springer, New York, Aug. 1990, pp. 239–252. DOI: 10.1007/0-387-34805-0_22.