# 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

**KeyGen**

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

**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, $\sigma$, 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*

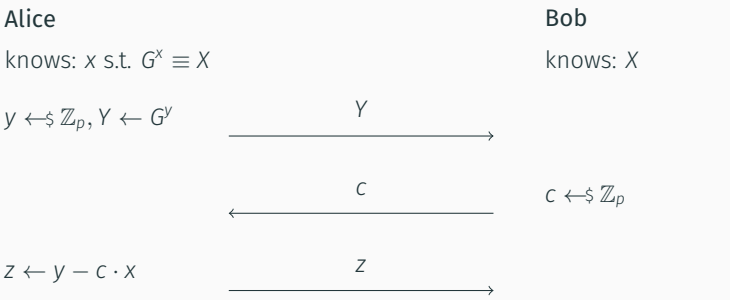
# Recap: Proof

*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



# 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 $c, c'$ for a fixed $Y$.

**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 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($X$) |
|---|---|
| 1 : $X \leftarrow G^x$ | 1 : |
| 2 : $y \leftarrow\!\!\$\ \mathbb{Z}_p$ | 2 : $c \leftarrow\!\!\$\ \mathbb{Z}_p$ |
| 3 : $Y \leftarrow G^y$ | 3 : $z \leftarrow\!\!\$\ \mathbb{Z}_p$ |
| 4 : $c \leftarrow\!\!\$\ \mathbb{Z}_p$ | 4 : $Y \leftarrow G^z \cdot X^{-c}$ |
| 5 : $z \leftarrow y - c \cdot x \bmod p$ | 5 : |
| 6 : **return** $(X, Y, c, z)$ | 6 : **return** $(X, Y, c, z)$ |

We want to show that the outputs of these two games are indistinguishable.

# Proof of Zero-Knowledge Property

| $\text{Game}_0(x)$ | $\text{Game}_1(x)$ | $\text{Game}_2(x)$ | $\text{Game}_3(x)$ |
|---|---|---|---|
| 1 : $X \leftarrow G^x$ | 1 : $X \leftarrow G^x$ | 1 : $X \leftarrow G^x$ | 1 : $X \leftarrow G^x$ |
| 2 : $y \leftarrow\$ \mathbb{Z}_p$ | 2 : $c \leftarrow\$ \mathbb{Z}_p$ | 2 : $c \leftarrow\$ \mathbb{Z}_p$ | 2 : $c \leftarrow\$ \mathbb{Z}_p$ |
| 3 : $Y \leftarrow G^y$ | 3 : $y \leftarrow\$ \mathbb{Z}_p$ | 3 : $z \leftarrow\$ \mathbb{Z}_p$ | 3 : $z \leftarrow\$ \mathbb{Z}_p$ |
| 4 : $c \leftarrow\$ \mathbb{Z}_p$ | 4 : $z \leftarrow y - c \cdot x$ | 4 : $y \leftarrow z + c \cdot x$ | 4 : |
| 5 : $z \leftarrow y - c \cdot x$ | 5 : $Y \leftarrow G^y$ | 5 : $Y \leftarrow G^y$ | 5 : $Y \leftarrow G^z \cdot X^c$ |
| 6 : **return** $(X, Y, c, z)$ | 6 : **return** $(X, Y, c, z)$ | 6 : **return** $(X, Y, c, z)$ | 6 : **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 $\varepsilon$ 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 $\varepsilon$?

# 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 $\mathbf{H}$.

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

# The Kernel of the Argument II

We know that the fraction of "1" entries in $\mathbf{H}$ is $\varepsilon$ 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 $\mathbf{H}$ into $\mathbf{H}_h$ with heavy rows and $\mathbf{H}_\ell$ with the remaining rows (i.e. fewer "1"s than a fraction of $\varepsilon/2$.)

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

# 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**

knows: $x$ s.t. $G^x \equiv X$

$y \leftarrow\!\!\$\ \mathbb{Z}_p, Y \leftarrow G^y$

$c \leftarrow \mathrm{H}(Y)$

$z \leftarrow y - c \cdot x$ $\xrightarrow{\quad Y, z \quad}$

**Bob**

knows: $X$

$c \leftarrow \mathrm{H}(Y)$

**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

**KeyGen**

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

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. $z \leftarrow y - c \cdot x$

$\sigma := (Y, z)$

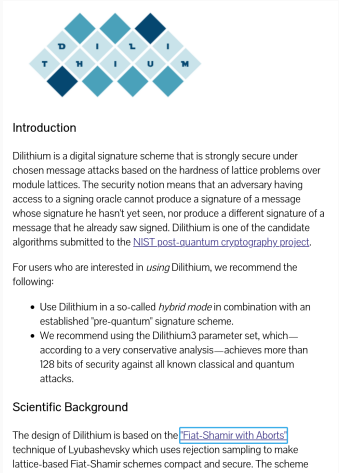
**Verify(vk, $\sigma$, m)**

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



DILITHIUM

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 $y_0 - y_1$ 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 $y_0 - y_1$ 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 I (Soundness)

# 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

## 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

## Post-Quantum Lattice-Based Version

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

# Interactive Protocol I

Given $G_0, G_1, Y_0, Y_1$

| Alice | | Bob |
|---|---|---|
| knows $G_0^x = Y_0$ and $G_1^x = Y_1$ | | |
| $y \leftarrow\$ \mathbb{Z}_p, A_0 \leftarrow G^y, A_1 \leftarrow G_1^y$ | $A_0, A_1 \longrightarrow$ | |
| | $\longleftarrow c$ | $c \leftarrow\$ \mathbb{Z}_p$ |
| $z \leftarrow y - c \cdot x$ | $z \longrightarrow$ | |
| | | assert $A_0 \equiv G_0^z \cdot Y_0^c \wedge A_1 \equiv G_1^z \cdot Y_1^c$ |

# Completeness (Correctness)

We check that $A_i$ equals $G_i^z \cdot Y_i^c$. Let's plug in:

$$\begin{aligned} A_i = G_i^y \stackrel{?}{\equiv} G_i^z \cdot Y_i^c \equiv G_i^{y-c\cdot x} \cdot Y_i^c \equiv G_i^{y-c\cdot x} \cdot (G_i^x)^c \\ \equiv G_i^{y-c\cdot x} \cdot G_i^{c\cdot x} \equiv G_i^{y-c\cdot x+c\cdot x} \equiv G_i^y \end{aligned}$$

# 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 $Y_0 = G_0^x$ and $Y_1 = G_1^x$ 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 $\mathsf{H}_0()$ and $\mathsf{H}_1()$.
- We assume there is some generator $G_0 \in \mathbb{G}$. Pick a random $G_1 \in \mathbb{G}$.

**KeyGen** $x \leftarrow\!\!\$\ \mathbb{Z}_p$, set $Y_0,\ Y_1 \leftarrow G_0^x,\ G_1^x$, $\mathsf{vk} \coloneqq (Y_0, Y_1)$ and $\mathsf{sk} \coloneqq x$.

**Sign**
1. Compute $y \leftarrow \mathsf{H}_0(x, m)$
2. Compute $A_0,\ A_1 \leftarrow G_0^y,\ G_1^y$
3. Compute $c \leftarrow \mathsf{H}_1(Y_0, Y_1, A_0, A_1, m)$.
4. Compute $z = y - c \cdot x \bmod p$ and return $\sigma \coloneqq (c, z)$.

**Verify**
1. Compute $A'_0,\ A'_1 \leftarrow G_0^z \cdot Y_0^c,\ G_1^z \cdot Y_1^c$.
2. Accept if $c \stackrel{?}{=} \mathsf{H}_1(Y_0, Y_1, A'_0, A'_1, m)$ otherwise reject.

# 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: 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 $H_1()$ for that
  - This is the same strategy for proving zero-knowledge as before with rewinding
- We don't really use $H_0()$ in the proof except for calling it.

# 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: SUF-CMA

| SUF-CMA | S($m$) | H($m$) |
|---|---|---|
| $\mathcal{Q}, \mathcal{H} \leftarrow \emptyset, \emptyset$; | $\sigma \leftarrow\$ \Sigma.\mathsf{Sign}(\mathsf{sk}, m)$ | **if** $m \notin \mathcal{H}$ **then** |
| $\mathsf{vk}, \mathsf{sk} \leftarrow\$ \Sigma.\mathsf{KeyGen}(1^\lambda)$; | $\mathcal{Q} \leftarrow \mathcal{Q} \cup \{(m, \sigma)\}$ | $\mathcal{H}[m] \leftarrow\$ \{0,1\}^\lambda$ |
| $(m^\star, \sigma^\star) \leftarrow\$ \mathcal{A}^{\mathsf{S},\mathsf{H}}(\mathsf{vk})$; | **return** $\sigma$ | **return** $\mathcal{H}[m]$ |
| $b_0 := (m^\star, \sigma^\star) \notin \mathcal{Q}$ | | |
| $b_1 := \Sigma.\mathsf{Verify}(\mathsf{vk}, \sigma^\star, m^\star) = 1$ | | |
| **return** $b_0 \wedge b_1$ | | |

$$\mathsf{Adv}^{\text{euf-cma}}_{\mathcal{A},\Sigma}(\lambda) := \Pr[\text{EUF-CMA}^{\mathcal{A}}_{\Sigma}(\lambda) \Rightarrow 1]$$

# Proof: Instantiating the Scheme

| $\mathsf{Game}_0$ | $\mathsf{S}(m)$ | $\mathsf{H}(m)$ |
|---|---|---|
| $\mathcal{Q}, \mathcal{H} \leftarrow \emptyset, \emptyset$ | $y \leftarrow \mathsf{H}_0(x, m)$ | **if** $m \notin \mathcal{H}$ **then** |
| $x \leftarrow\!\!\$\ \mathbb{Z}_p;\ Y_0,\ Y_1 \leftarrow G_0^x,\ G_1^x$ | $A_0, A_1 \leftarrow G_0^y, G_1^y$ | $\mathcal{H}[m] \leftarrow\!\!\$\ \mathbb{Z}_p$ |
| $\mathsf{vk},\ sk := (Y_0, Y_1),\ x$ | $c \leftarrow \mathsf{H}_1(Y_0, Y_1, A_0, A_1, m)$ | **return** $\mathcal{H}[m]$ |
| $(m^\star, \sigma^\star) \leftarrow\!\!\$\ \mathcal{A}^{\mathsf{S},\mathsf{H}}(\mathsf{vk})$ | $z = y - c \cdot x \bmod p$ | |
| $b_0 := (m^\star, \sigma^\star) \notin \mathcal{Q}$ | $\mathcal{Q} \leftarrow \mathcal{Q} \cup \{(m, (c, z))\}$ | |
| $(c^\star, z^\star) := \sigma^\star$ | **return** $(c, z)$ | |
| $A_0^\star, A_1^\star \leftarrow G_0^{z^\star} \cdot Y_0^{c^\star}, G_1^{z^\star} \cdot Y_1^{c^\star}$ | | |
| $b_1 := \mathsf{H}_1(Y_0, Y_1, A_0^\star, A_0^\star, m^\star) \stackrel{?}{=} c^\star$ | | |
| **return** $b_0 \wedge b_1$ | | |

# Proof: Return Known Signatures

| $\mathsf{Game}_1$ | $\mathsf{S}(m)$ | $\mathsf{H}(m)$ |
|---|---|---|
| $\mathcal{Q}, \mathcal{H} \leftarrow \emptyset, \emptyset$ | **if** $(m', \sigma') \in \mathcal{Q}$ s.t. $m = m'$ | **if** $m \notin \mathcal{H}$ **then** |
| $x \leftarrow\!\!\$\ \mathbb{Z}_p$; $Y_0, Y_1 \leftarrow G_0^x, G_1^x$ | **return** $\sigma'$ | $\mathcal{H}[m] \leftarrow\!\!\$\ \mathbb{Z}_p$ |
| $\mathsf{vk},\ \mathsf{sk} \coloneqq (Y_0, Y_1),\ x$ | $y \leftarrow \mathsf{H}_0(x, m)$ | **return** $\mathcal{H}[m]$ |
| $(m^\star, \sigma^\star) \leftarrow\!\!\$\ \mathcal{A}^{\mathsf{S},\mathsf{H}}(\mathsf{vk})$ | $A_0, A_1 \leftarrow G_0^y, G_1^y$ | |
| $b_0 \coloneqq (m^\star, \sigma^\star) \notin \mathcal{Q}$ | $c \leftarrow \mathsf{H}(Y_0, Y_1, A, B, m)$ | |
| $(c^\star, z^\star) \coloneqq \sigma^\star$ | $z = y - c \cdot x \mod p$ | |
| $A_0^\star,\ A_1^\star \leftarrow G_0^{z^\star} \cdot Y_0^{c^\star},\ G_1^{z^\star} \cdot Y_1^{c^\star}$ | $\mathcal{Q} \leftarrow \mathcal{Q} \cup \{(m, (c, z))\}$ | |
| $b_1 \coloneqq \mathsf{H}_1(Y_0, Y_1, A_0^\star, A_1^\star, m^\star) \stackrel{?}{=} c^\star$ | **return** $(c, z)$ | |
| $b_0 \wedge b_1$ | | |

No change in behaviour.

# Proof: Simulate the Signing Oracle I

| $\mathrm{Game}_2$ | $\mathrm{S}(m)$ | $\mathrm{H}(m)$ |
|---|---|---|
| $\mathcal{Q}, \mathcal{H} \leftarrow \emptyset, \emptyset$ | **if** $(m', \sigma') \in \mathcal{Q}$ s.t. $m = m'$ | **if** $m \notin \mathcal{H}$ **then** |
| bad $\leftarrow$ **false** | **return** $\sigma'$ | $\mathcal{H}[m] \leftarrow\!\$\ \mathbb{Z}_p$ |
| $x \leftarrow\!\$\ \mathbb{Z}_p$; $Y_0, Y_1 \leftarrow G_0^x, G_1^x$ | $c, z \leftarrow\!\$\ \mathbb{Z}_p^2$ | **return** $\mathcal{H}[m]$ |
| $\mathsf{vk},\ \mathsf{sk} := (Y_0, Y_1),\ x$ | $A_0, A_1 \leftarrow G_0^z \cdot Y_0^{-c},\ G_1^z \cdot Y_1^{-c}$ | |
| $(m^\star, \sigma^\star) \leftarrow\!\$\ \mathcal{A}^{\mathrm{S},\mathrm{H}}(\mathsf{vk})$ | **if** $(Y_0, Y_1, A_0, A_1, m) \in \mathcal{H}$ | |
| $b_0 := (m^\star, \sigma^\star) \notin \mathcal{Q}$ | bad $\leftarrow$ **true** | |
| $(c^\star, z^\star) := \sigma^\star$ | **abort** | |
| $A_0^\star,\ A_1^\star \leftarrow G_0^{z^\star} \cdot Y_0^{c^\star},\ G_1^{z^\star} \cdot Y_1^{c^\star}$ | $\mathcal{H}[(Y_0, Y_1, A_0, A_1, m)] \leftarrow c$ | |
| $b_1 := \mathrm{H}_1(Y_0, Y_1, A_0^\star, A_1^\star, m^\star) \stackrel{?}{=} c^\star$ | $\mathcal{Q} \leftarrow \mathcal{Q} \cup \{(m, (c, z))\}$ | |
| $b_0 \wedge b_1$ | **return** $(c, z)$ | |

Fundamental Lemma of Game Playing and programming the RO

# Proof: Simulate the Signing Oracle II

- 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 $\mathsf{H}_1(Y_0, Y_1, A_0, A_1, m)$ before.
- This means it has guessed $y$ correctly to compute $A_0 = G_0^y$ and $A_1 = G_1^y$.
- We can show that this probaliltiy is exponentially small, but I'll avoid the details here.

# Proof: Making this Adversary Useful

$\mathsf{Game}_3(G_0, G_1, Y_0, Y_1)$

$\mathcal{Q}, \mathcal{H} \leftarrow \emptyset, \emptyset$

$\mathsf{bad} \leftarrow \textbf{false}$

$\mathsf{vk},\ \mathsf{sk} := (Y_0, Y_1),\ \perp$

$(m^\star, \sigma^\star) \leftarrow\!\$\ \mathcal{A}^{\mathsf{S},\mathsf{H}}(\mathsf{vk})$

$b_0 := (m^\star, \sigma^\star) \notin \mathcal{Q};\ (c^\star, z^\star) := \sigma^\star$

$A_0^\star,\ A_1^\star \leftarrow G_0^{z^\star} \cdot Y_0^{c^\star},\ G_1^{z^\star} \cdot Y_1^{c^\star};$

$b_1 := \mathsf{H}_1(Y_0, Y_1, A_0^\star, A_1^\star, m^\star) \stackrel{?}{=} c^\star$

$b_0 \wedge b_1$

$\mathsf{S}(m)$

**if** $(m', \sigma') \in \mathcal{Q}$ s.t. $m = m'$

  **return** $\sigma'$

$c, z \leftarrow\!\$\ \mathbb{Z}_p^2$

$A_0, A_1 \leftarrow G_0^z \cdot Y_0^{-c},\ G_1^z \cdot Y_1^{-c}$

**if** $(Y_0, Y_1, A_0, A_1, m) \in \mathcal{H}$

  $\mathsf{bad} \leftarrow \textbf{true}$

  **abort**

$\mathcal{H}[(Y_0, Y_1, A_0, A_1, m)] \leftarrow c$

$\mathcal{Q} \leftarrow \mathcal{Q} \cup \{(m, (c, z))\}$

**return** $(c, z)$

$\mathsf{H}(m)$

**if** $m \notin \mathcal{H}$ **then**

  $\mathcal{H}[m] \leftarrow\!\$\ \mathbb{Z}_p$

**return** $\mathcal{H}[m]$

The adversary **cannot win** if our input $(G_0, G_1, Y_0, Y_1)$ is not a DH tuple.

# Proof: Solving DDH I

**Lemma**

*Assume there is no $x$ s.t. $G_0^x = Y_0$ and $G_1^x = Y_1$. Then for any $A_0, A_1$ output by $\mathcal{A}$ there is at most one value of $c$ for which the game will accept.*

**Proof.**

Say $A_0, A_1 \in \mathbb{G}$ are such that the signer can send correct responses $z_0, z_1$ for two different challenges $c_0, c_1$. Then

$$A_0 = G_0^{z_0} \cdot Y_0^{c_0} = G_0^{z_1} \cdot Y_0^{c_1} \text{ and } A_1 = G_1^{z_0} \cdot Y_1^{c_0} = G_1^{z_1} \cdot Y_1^{c_1}.$$

Noting that $c_1 \neq c_0$ we have

$$Y_0 = G_0^{(z_0 - z_1)/(c_1 - c_0)} \text{ and } G_1^{(z_0 - z_1)/(c_1 - c_0)} = Y_1$$

contrary to the assumption. □

# Proof: Solving DDH II

When

$$(G, G^y, G^x, Z) = (G, G^y, G^x, G^{xy}) = (G_0, G_1, Y_0, Y_1) = (G_0, G_1, G_0^x, G_1^x)$$

then the adversary "lives" exactly in the world where it succeeds, i.e. it will output a forgery (with some probability not discussed here).

# Proof: Solving DDH III

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": $A_0 = G_0^z \cdot Y_0^c$ and $A_1 = G_1^z \cdot Y_1^c$.
- But $c$ is the output of a random oracle, i.e. random.
- The probability of hitting that one magical $c$ where a solution exists is negligible.
- In other words, whatever the adversary does it cannot win (except with very low probability).

# 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.

# Avoiding Rewinding II (Fischlin Transform)

# Main Reference

Marc Fischlin. **Communication-Efficient Non-interactive Proofs of Knowledge with Online Extractors.** In: *CRYPTO 2005*. Ed. by Victor Shoup. Vol. 3621. LNCS. Springer, Berlin, Heidelberg, Aug. 2005, pp. 152–168. DOI: 10.1007/11535218_10

# Recap: Special Soundness

If Alice answers for at least two different challenges $c, c'$ for a fixed $Y$, we can extract a solution.

**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. $G^{(z-z')/(c'-c)} \equiv X$

# Recap: Special Soundness

If Alice answers for at least two different challenges $c, c'$ for a fixed $Y$, we can extract a solution.

**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. $G^{(z-z')/(c'-c)} \equiv X$

**Task**

Somehow convince Alice to submit both $(X, Y, c, z)$ and $(X, Y, c', z')$ to a Random Oracle.

# The Scheme (Simplified)

KeyGen($1^\lambda$): $\mathsf{sk} := x \leftarrow\!\!\$\ \mathbb{Z}_p$; $\mathsf{vk} := X \leftarrow G^x$; **return** vk, sk

Sign(sk, $m$)

1: **for** $0 \le i < r$ **do**
2: $\quad y_i \leftarrow\!\!\$\ \mathbb{Z}_p$; $Y_i \leftarrow G^{y_i}$
3: **for** $0 \le i < r$ **do**
4: $\quad$ **while true**
5: $\quad\quad c_i \leftarrow\!\!\$\ \mathbb{Z}_p$; $z_i \leftarrow y_i - c_i \cdot x$
6: $\quad\quad h_i \leftarrow \mathsf{H}(X, \{Y_i\}_{0 \le i < r}, m, i, c_i, z_i)$
7: $\quad\quad$ **if** $h_i \bmod 2^b \equiv 0$ **then break**
8: **return** $\sigma := \{(Y_i, c_i, z_i)\}_{0 \le i < r}$

Verify(vk, $\sigma = \{(Y_i, c_i, z_i)\}, m$)

1: **for** $0 \le i < r$ **do**
2: $\quad$ **if** $Y_i \neq G^{z_i} \cdot X^{c_i}$ **then**
3: $\quad\quad$ **return false**
4: $\quad h_i \leftarrow \mathsf{H}(X, \{Y_i\}, m, i, c_i, z_i)$
5: $\quad$ **if** $h_i \bmod 2^b \neq 0$ **then**
6: $\quad\quad$ **return false**
7: **return true**

# Explanation

- The code $h_i \bmod 2^b \stackrel{?}{\equiv} 0$ checks whether the $b$ least significant bits are zero
  - just like the proof-of-work (PoW) in Bitcoin, but in contrast to PoW, we're actually exploiting this for something useful
- With probability $1 - 2^{-b}$ the prover needs to call $\mathsf{H}(X, \{Y_i\}_{0 \leq i < r}, m, i, c_i, z_i)$ more than once to obtain the desired result
  - We can leverage that for special soundness!
- We need to repeat this $r$ times, to enforce a malicious prover cannot get lucky

# "Knowledge Soundness" ≠ "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.

# References II


[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.

[Fis05] Marc Fischlin. Communication-Efficient Non-interactive Proofs of Knowledge with Online Extractors. In: *CRYPTO 2005*. Ed. by Victor Shoup. Vol. 3621. LNCS. Springer, Berlin, Heidelberg, Aug. 2005, pp. 152–168. DOI: 10.1007/11535218_10.

[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.

# References III


[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.