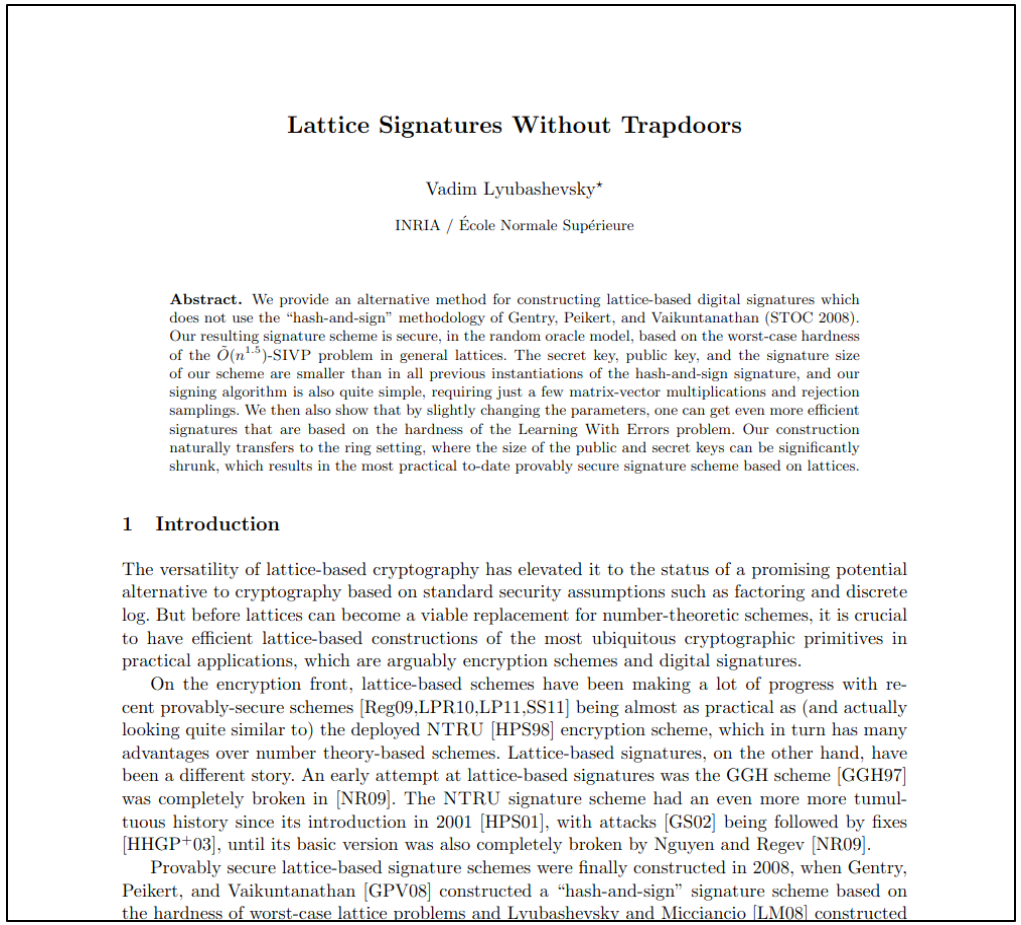
Fiat-Shamir with Aborts

Andreas Hülsing
Eindhoven University of Technology & SandboxAQ

Fiat-Shamir with Aborts

(Lyubashevsky, Eurocrypt 2012)

- Technique to build Schnorr-style signatures from lattices
- Uses Fiat-Shamir like Schnorr
- Difference: Requires "rejection sampling"
- Underlying BLISS, Dilithium, and many more



Lattice Signatures Without Trapdoors

Vadim Lyubashevsky*

INRIA / École Normale Supérieure

Abstract. We provide an alternative method for constructing lattice-based digital signatures which does not use the "hash-and-sign" methodology of Gentry, Peikert, and Vaikuntanathan (STOC 2008). Our resulting signature scheme is secure, in the random oracle model, based on the worst-case hardness of the $\tilde{O}(n^{1.5})$-SIVP problem in general lattices. The secret key, public key, and the signature size of our scheme are smaller than in all previous instantiations of the hash-and-sign signature, and our signing algorithm is also quite simple, requiring just a few matrix-vector multiplications and rejection samplings. We then also show that by slightly changing the parameters, one can get even more efficient signatures that are based on the hardness of the Learning With Errors problem. Our construction naturally transfers to the ring setting, where the size of the public and secret keys can be significantly shrunk, which results in the most practical to-date provably secure signature scheme based on lattices.

1 Introduction

The versatility of lattice-based cryptography has elevated it to the status of a promising potential alternative to cryptography based on standard security assumptions such as factoring and discrete log. But before lattices can become a viable replacement for number-theoretic schemes, it is crucial to have efficient lattice-based constructions of the most ubiquitous cryptographic primitives in practical applications, which are arguably encryption schemes and digital signatures.

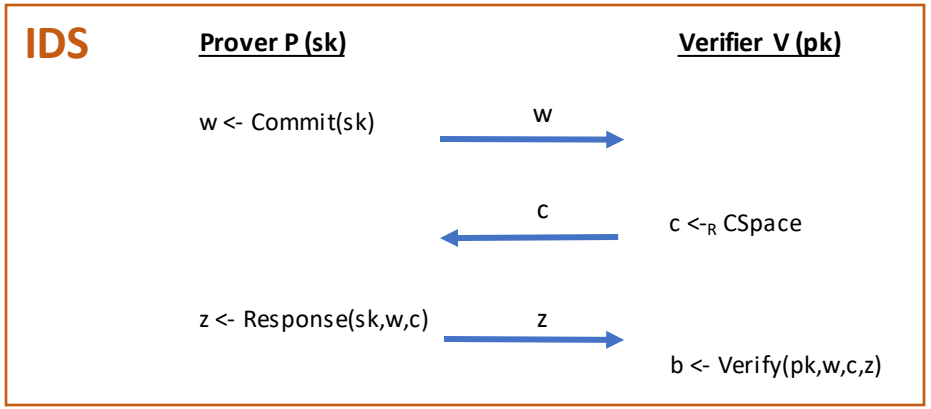
On the encryption front, lattice-based schemes have been making a lot of progress with recent provably-secure schemes [Reg09,LPR10,LP11,SS11] being almost as practical as (and actually looking quite similar to) the deployed NTRU [HPS98] encryption scheme, which in turn has many advantages over number theory-based schemes. Lattice-based signatures, on the other hand, have been a different story. An early attempt at lattice-based signatures was the GGH scheme [GGH97] was completely broken in [NR09]. The NTRU signature scheme had an even more more tumultuous history since its introduction in 2001 [HPS01], with attacks [GS02] being followed by fixes [HHGP+03], until its basic version was also completely broken by Nguyen and Regev [NR09].

Provably secure lattice-based signature schemes were finally constructed in 2008, when Gentry, Peikert, and Vaikuntanathan [GPV08] constructed a "hash-and-sign" signature scheme based on the hardness of worst-case lattice problems and Lyubashevsky and Micciancio [LM08] constructed

Outline

- Identification Schemes / Schnorr
- Fiat-Shamir: Signatures from Identification
- The case of Dilithium / Identification with abort
- A subtle proof-issue

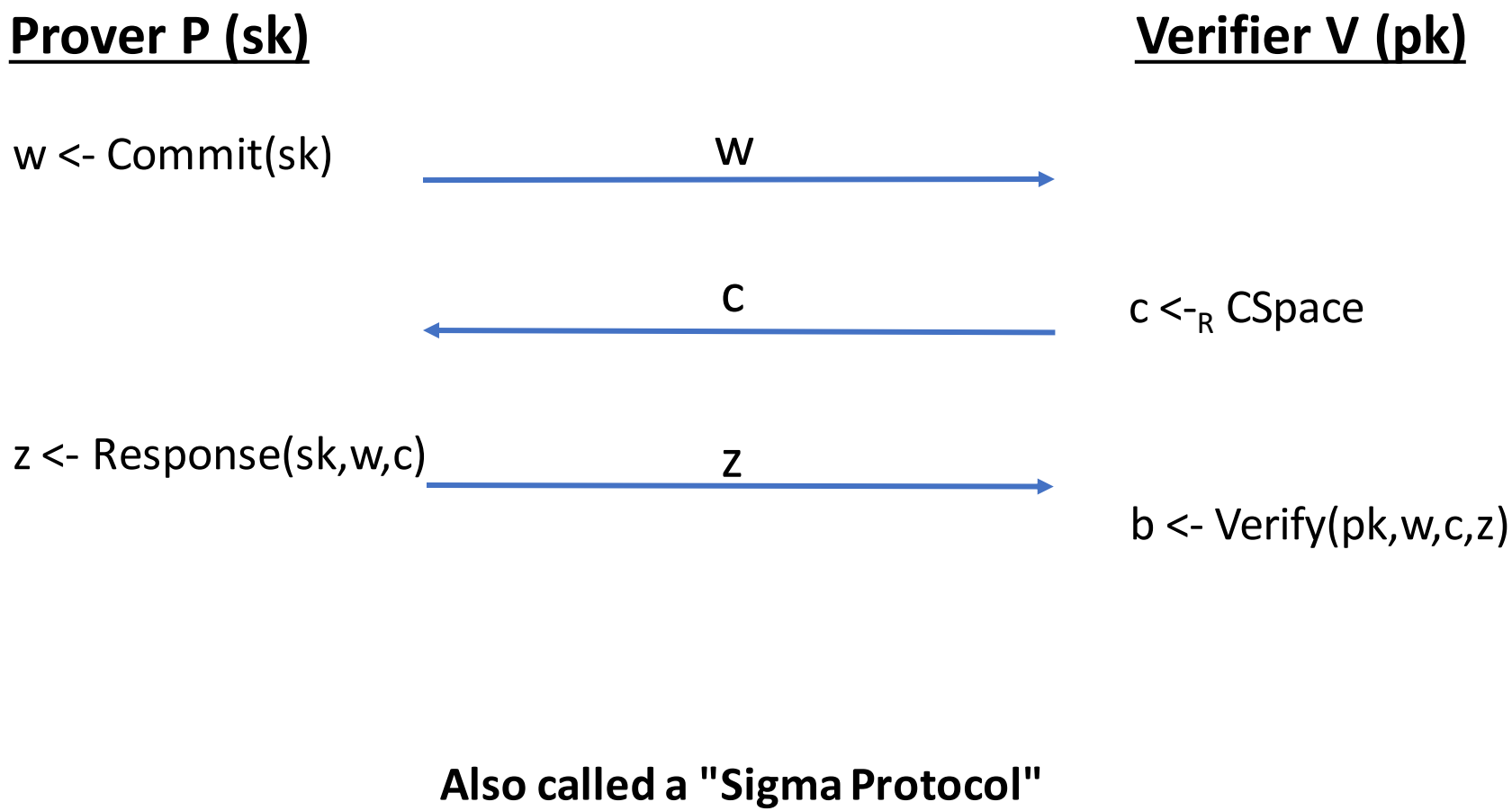
Identification Schemes



Identification Schemes (IDS) / Zero-knowledge proofs (ZKP)

- Invented by Shafi Goldwasser, Silvio Micali and Charles Rackoff in 1985
- Interactive proof systems
- Prove knowledge of a secret without revealing any information about the secret
- [For people that like classifications: The IDS we discuss are actually Honest-Verifier Zero-Knowledge Arguments of Knowledge]

Identification schemes (3-round, public coin)



Also called a "Sigma Protocol"

Commitment Scheme

- COM = (com, open)
  - w, x <- com(in; rand)
  - in' <- open(w, x)
- Binding: hard to find x ≠ x' s.th. open(w, x) ≠ open(w, x')
- Hiding: com(in; rand) ~ com(in'; rand')
- One-way: hard to learn in from w

- Examples:
- H(in;rand), rand <- com(in;rand) [hiding + binding]
- w <- g^{in} <- com(in) [one-way & binding]

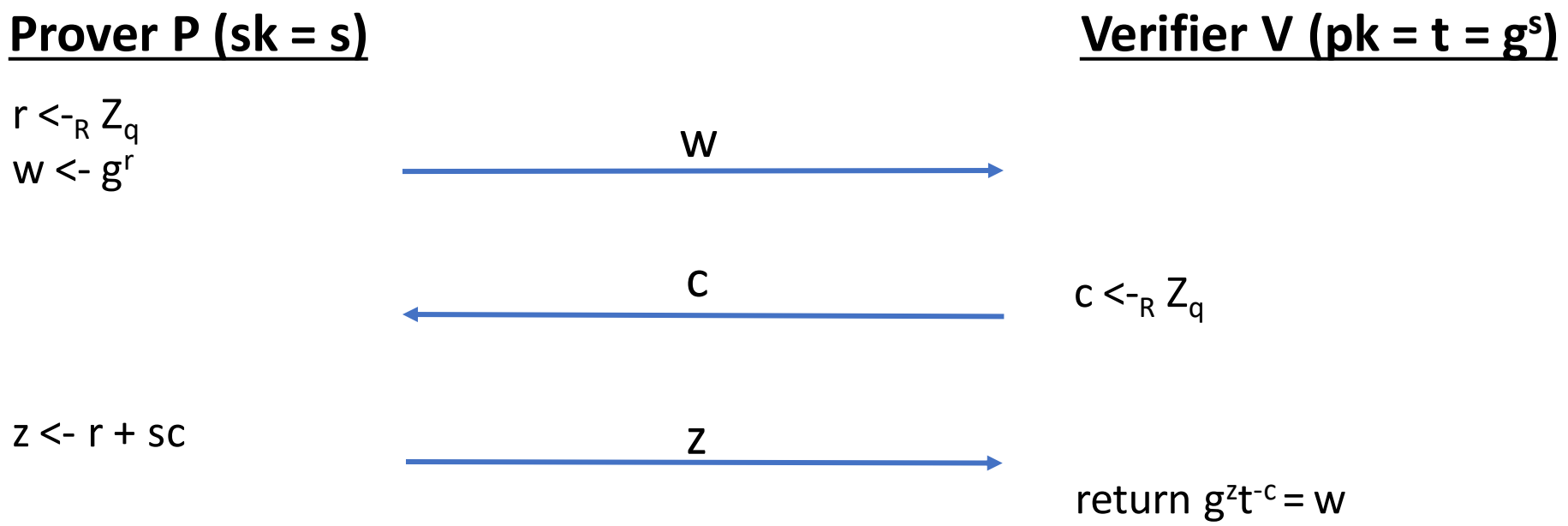
Security Properties

- **Soundness:** A prover that does not know the secret will get caught with high probability (1 – e) where e is called soundness error

- **Special soundness:** There exists an efficient extractor E that given two transcripts with same w but different c, extracts sk.

- **Honest verifier zero-knowledge (HVZK):** There exists an efficient simulator ZKSim that, given only the public key, outputs transcripts which are indistinguishable from transcripts of honest protocol runs

Schnorr Identification scheme



Correctness: $g^z t^{-c} = g^{r+sc} g^{(-cs)} = g^r = w$

Security - Soundness

Special soundness:

- Given valid ((w, c, z), (w, c', z')) we have (in Z_q)

$$z - z' = (r + sc) - (r + sc') = s(c-c')$$

and so

$$(z-z')(c-c')^{-1} = s$$

Soundness:

- Rewinding – If A has better than 1/q chance of winning per w, they must be able to respond to more than one c.

Security - HVZK

[Recall verify checks $g^z t^{-c} = w$]

Trick: Transcript does not enforce order

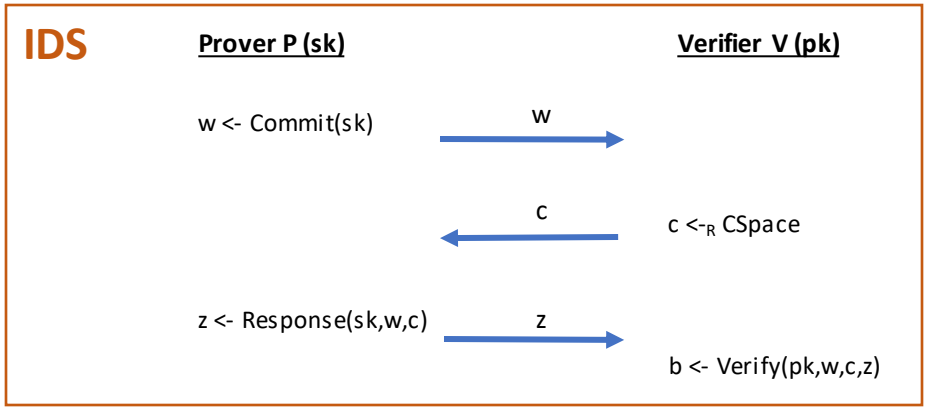
ZKSim(pk = t = q^s):

1. $c \leftarrow_R Z_q$
2. $z \leftarrow_R Z_q$
3. $w \leftarrow g^z t^{-c}$
4. Output(w, c, z)

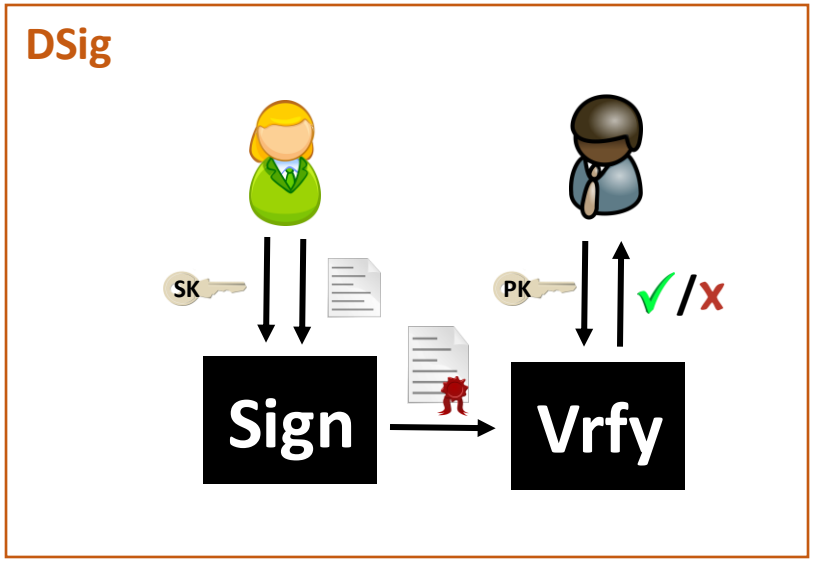
Trans(sk = s):

1. $r \leftarrow_R Z_q$
2. $w \leftarrow g^r$
3. $c \leftarrow_R Z_q$
4. $z \leftarrow r + sc$
5. Output(w, c, z)

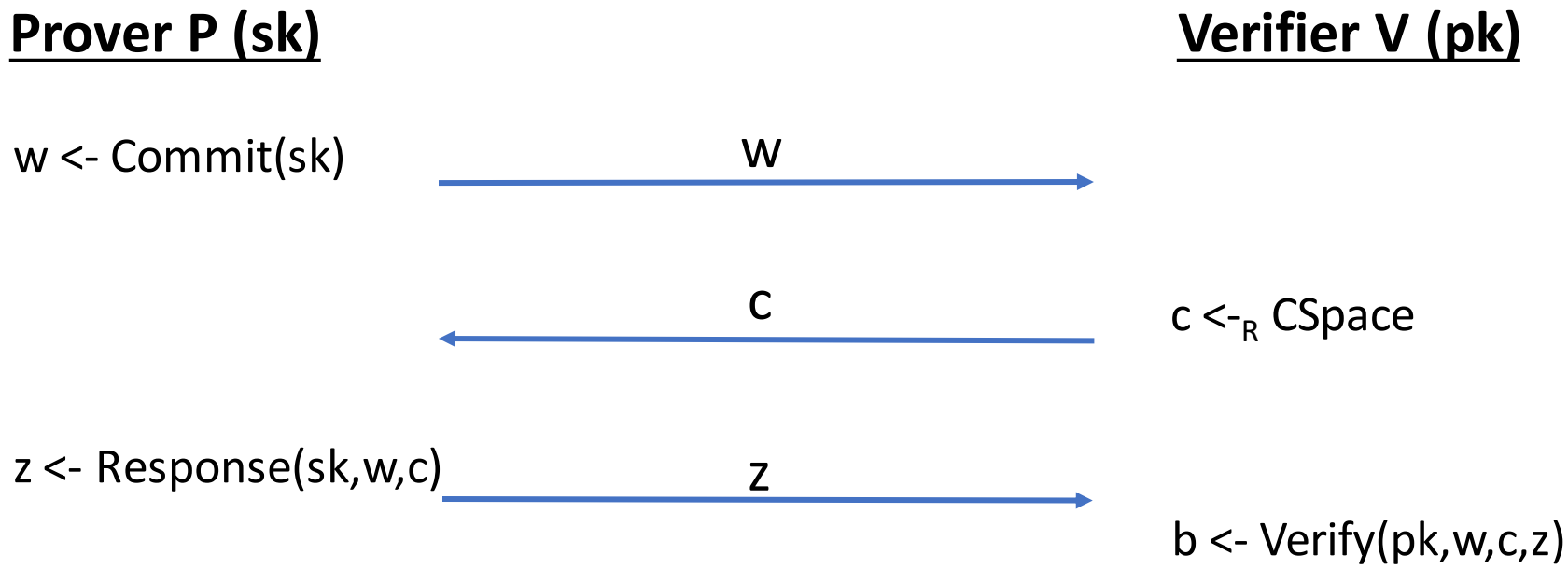
Fiat-Shamir



Fiat-Shamir



Identification schemes (3-round, public coin)



Fiat-Shamir Signatures

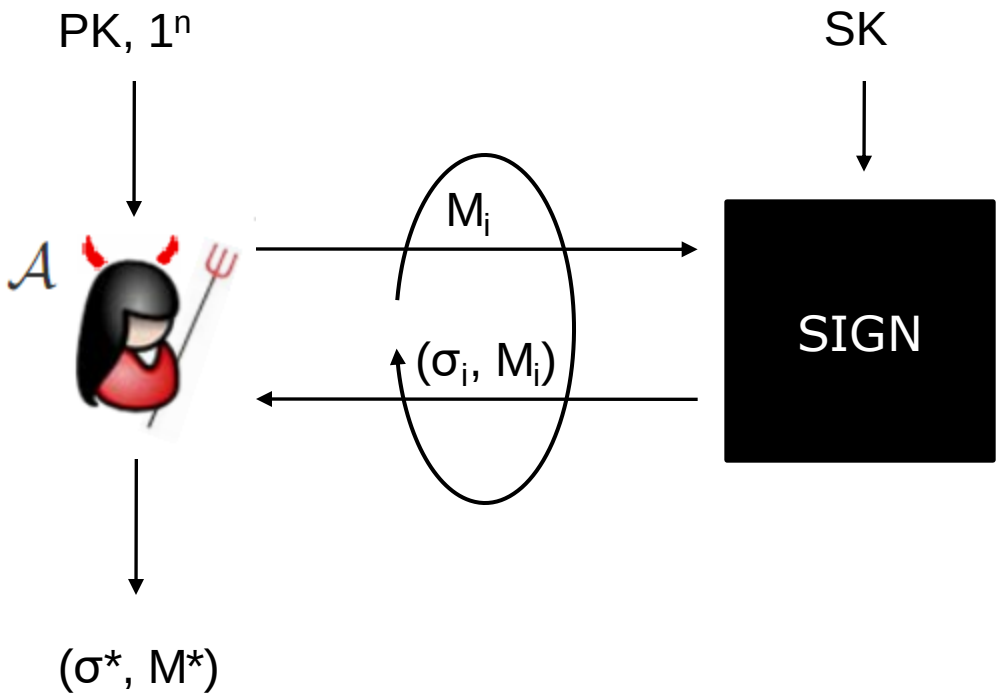
Sign<H> (sk,m)

1. w <- P.Commit(sk)
2. c <- H(pk, w, m)
3. z <- P.Response(sk, w, c)
4. Return sig = (w, z)

Verify<H> (pk, m, sig)

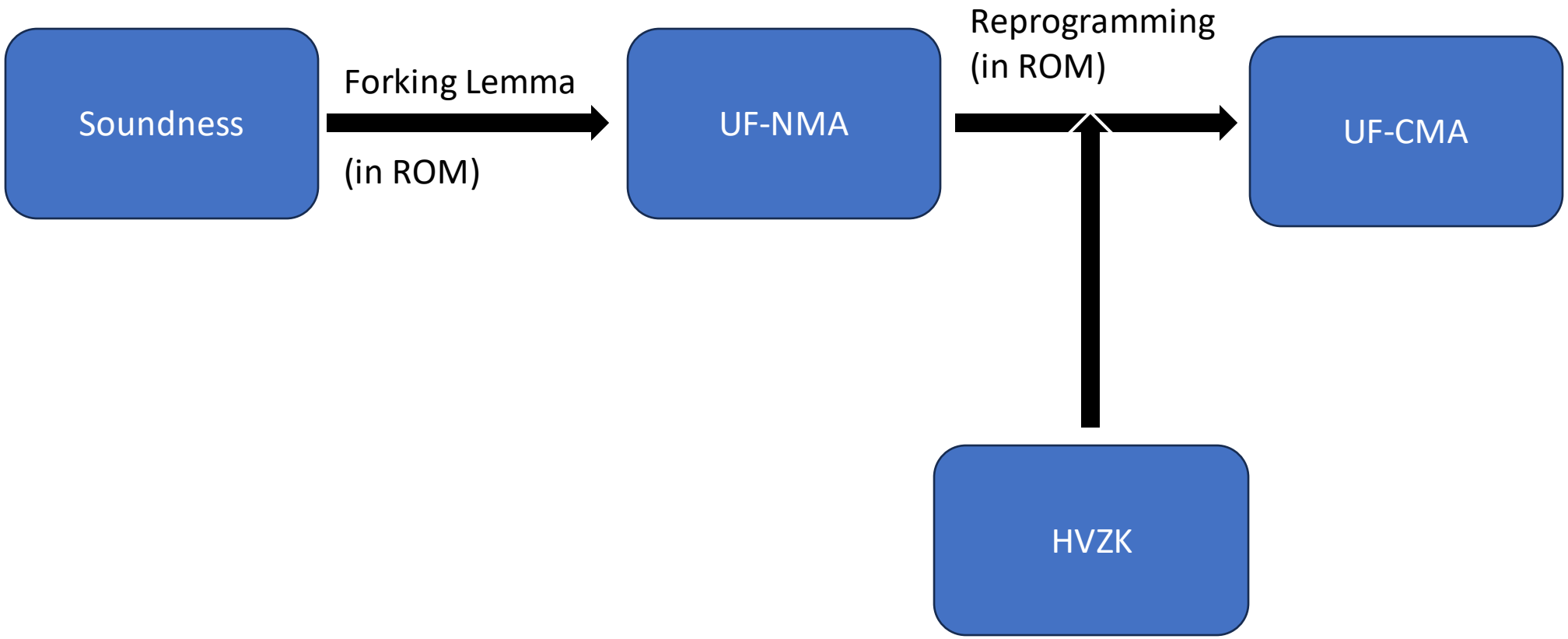
1. c <- H(pk, w, m)
2. b <- V.Verify(pk, w, c, z)

existential UnForgability under adaptive Chosen Message Attacks (UF-CMA)

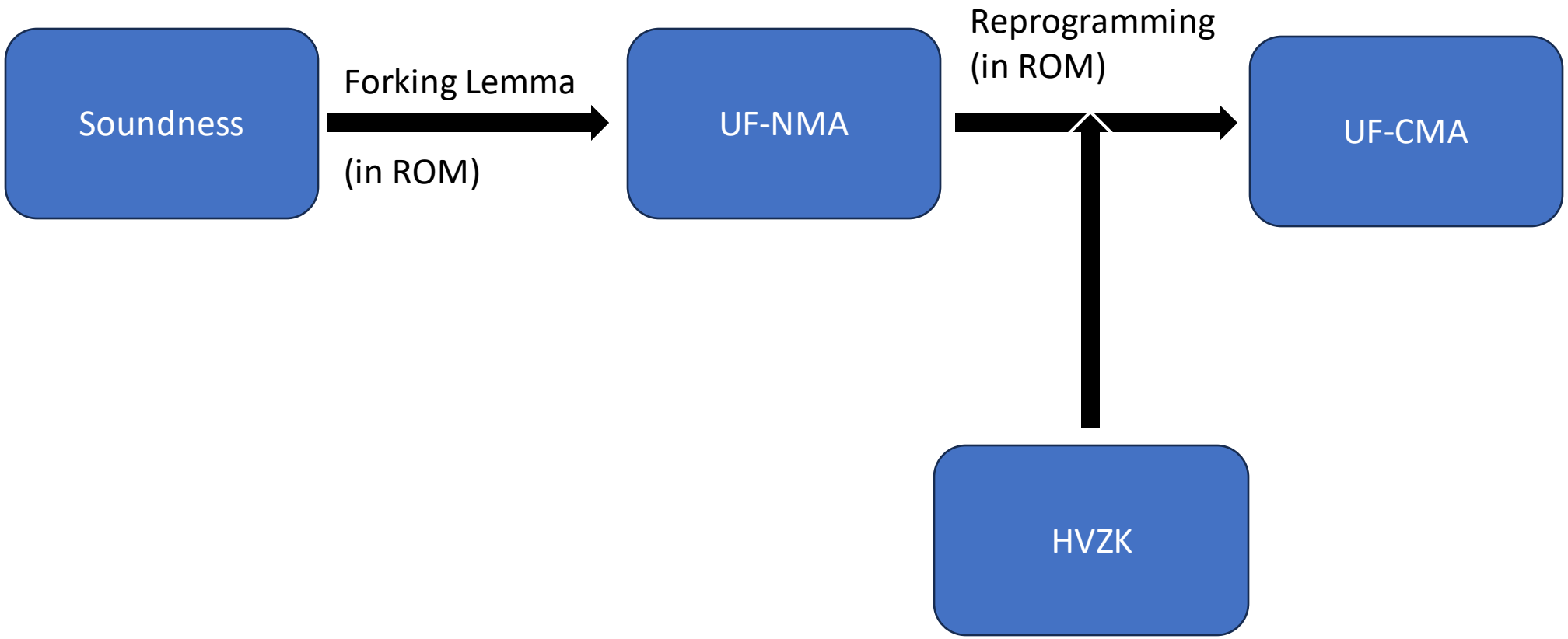


existential UnForgability under No-Message Attacks (UF-NMA)

Proof outline

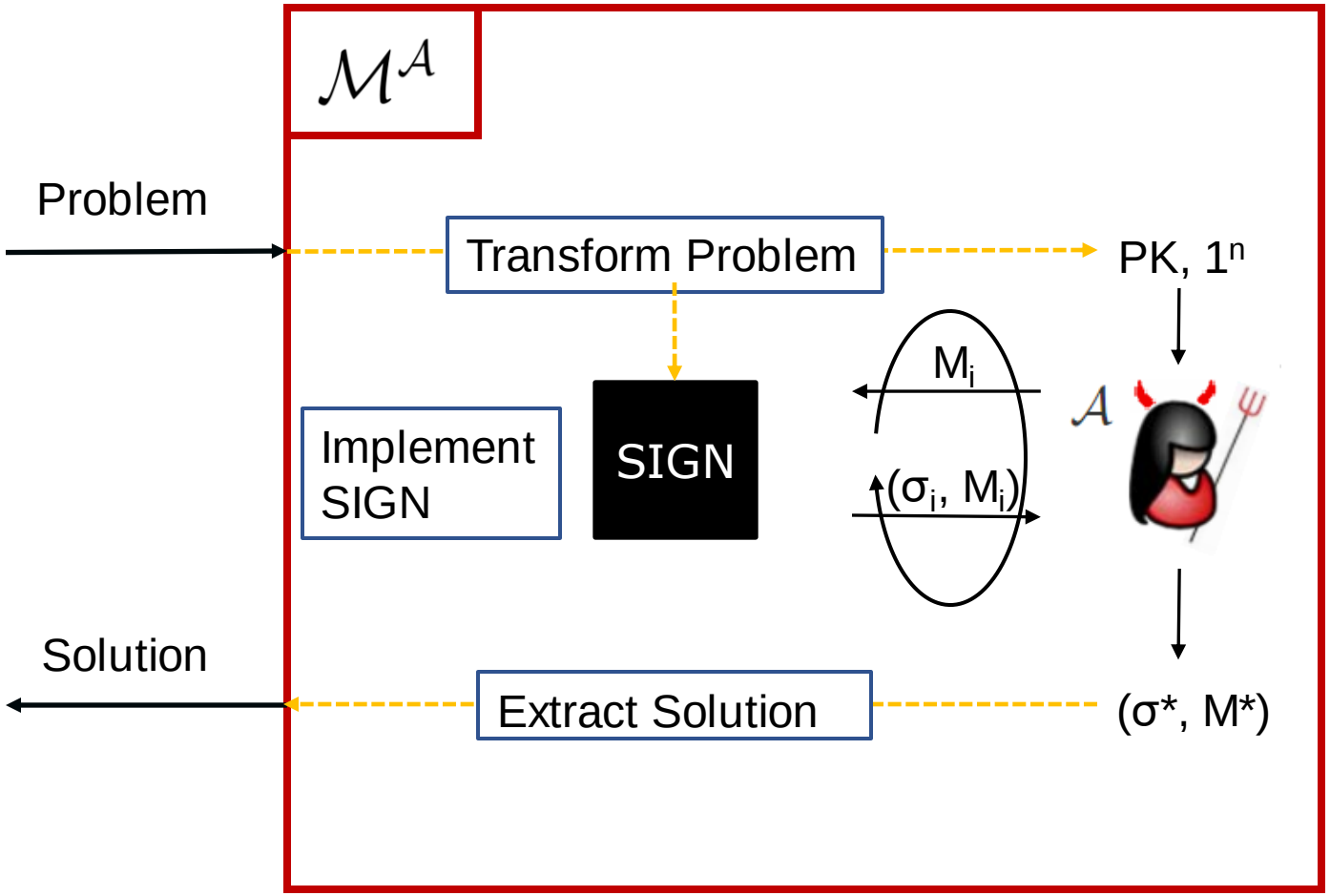


Proof outline

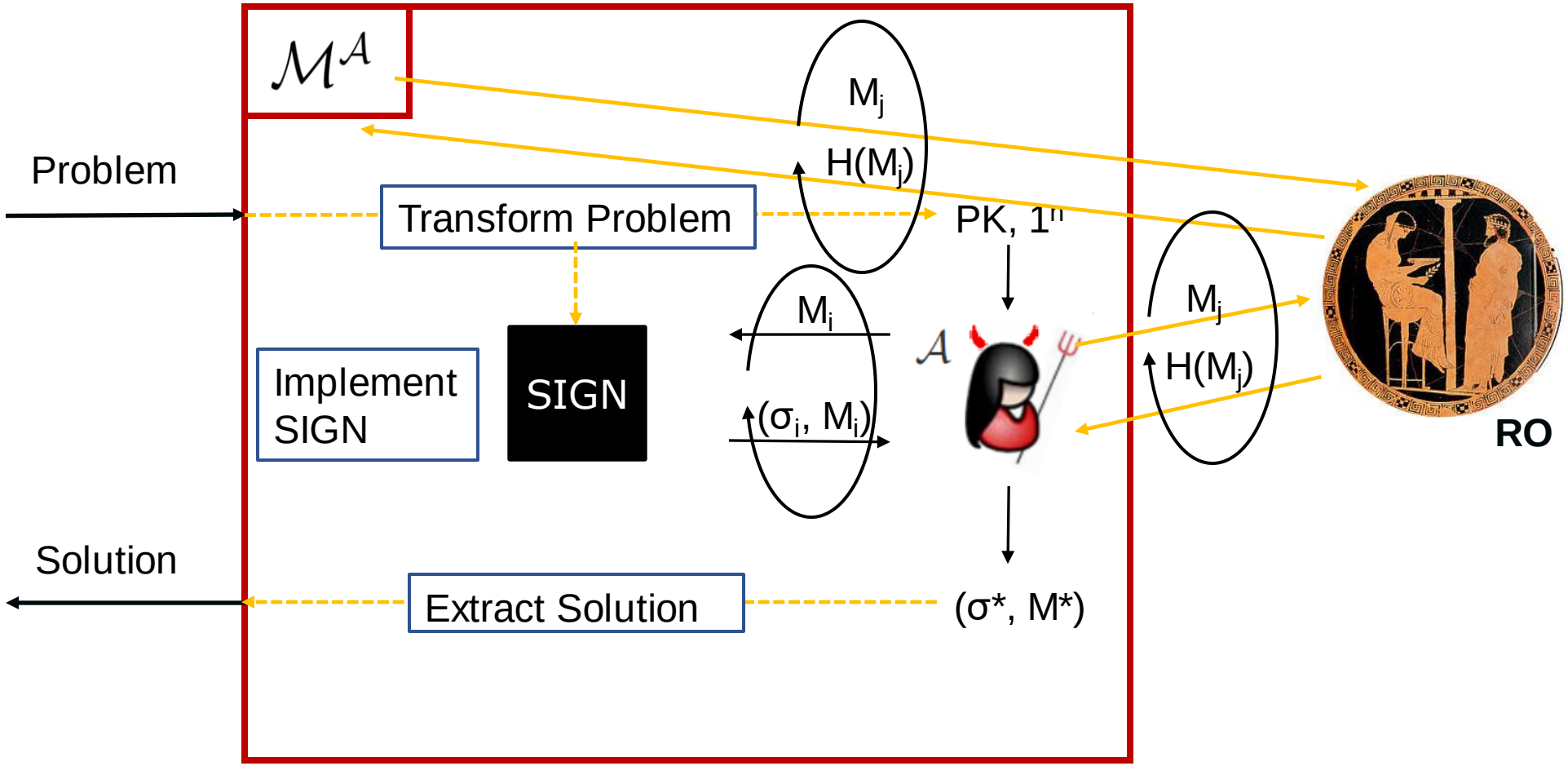


Indirection:
The random oracle model (ROM)

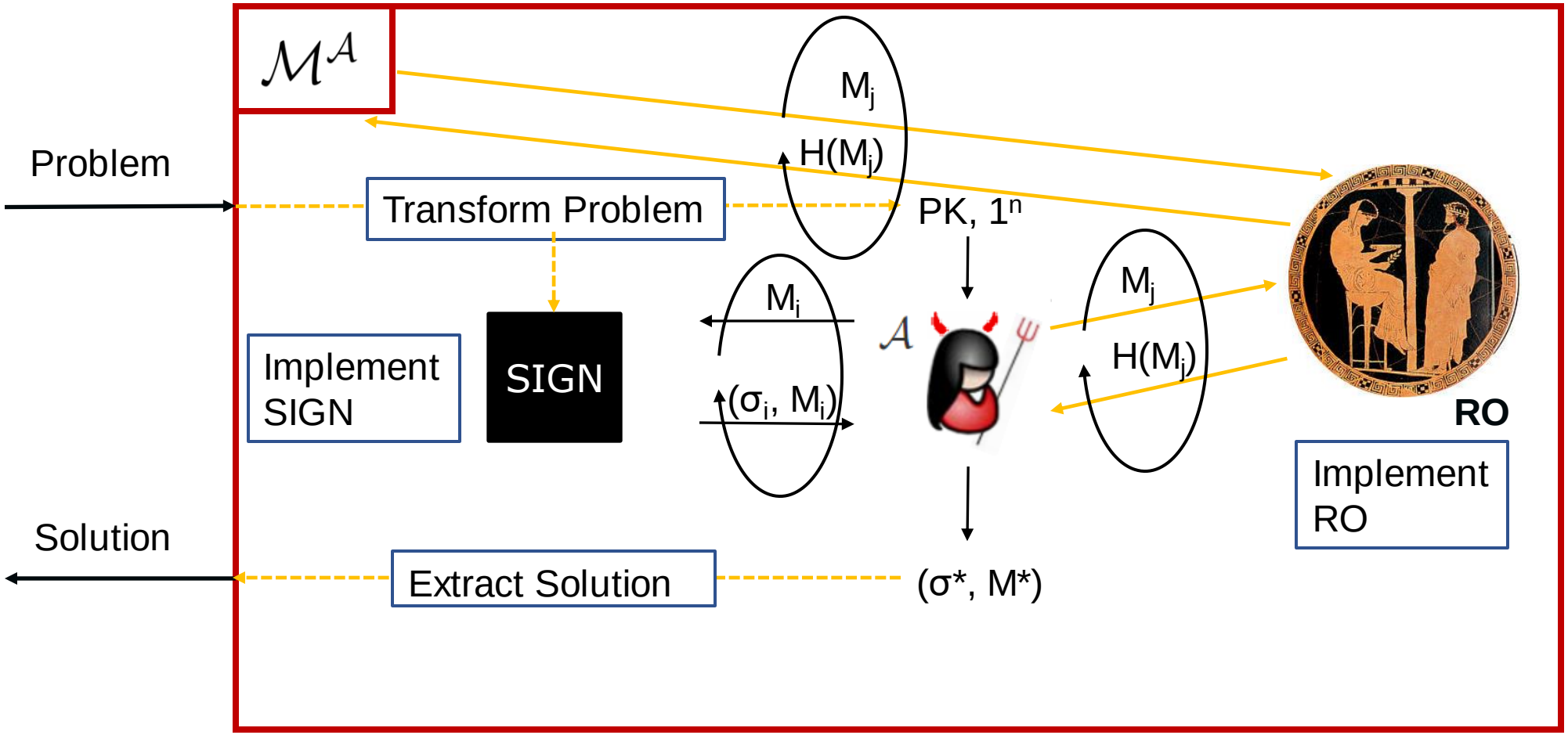
Reduction


$\mathcal{M}^{\mathcal{A}}$
Problem
Transform Problem
PK, 1^n
M_i
$\mathcal{A}$
Implement SIGN
SIGN
(σ_i, M_i)
Solution
Extract Solution
(σ^*, M^*)

Reduction in the random oracle model



Reduction in the random oracle model



RO Reprogramming

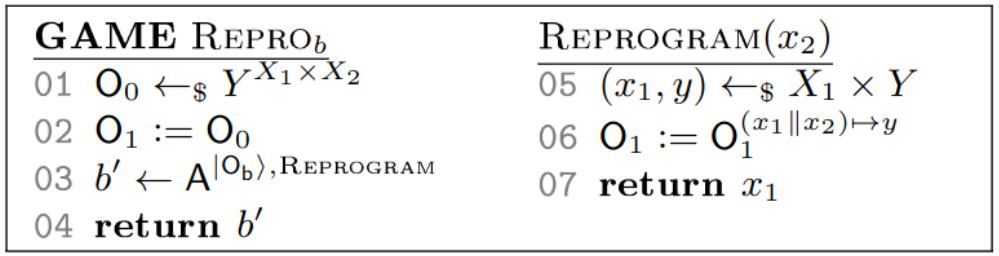


Fig. 1. Adaptive reprogramming games REPRO_b for bit $b \in \{0, 1\}$ in the most basic setting.

Proposition 1. *Let X_1, X_2 and Y be finite sets, and let A be any algorithm issuing R many calls to* REPROGRAM *and q many (quantum) queries to O_b as defined in Fig. 1. Then the distinguishing advantage of A is bounded by*

$$|\Pr[\text{REPRO}_1^{\mathsf{A}} \Rightarrow 1] - \Pr[\text{REPRO}_0^{\mathsf{A}} \Rightarrow 1]| \leq \frac{3R}{2}\sqrt{\frac{q}{|X_1|}}. \tag{1}$$

RO Reprogramming

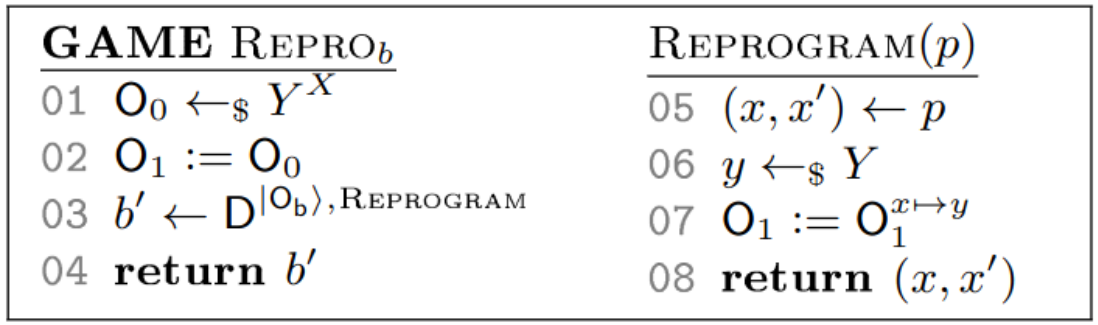


Fig. 2. Adaptive reprogramming games REPRO_b for bit $b \in \{0, 1\}$.

Proposition 2. *Let X_1, X_2, X' and Y be some finite sets, and let p be a distribution on $X_1 \times X'$. Let D be any distinguisher, issuing q many (quantum) queries to O and R many reprogramming instructions such that each instruction consists of a value x_2, together with the fixed distribution p. Then*

$$|\Pr[\text{REPRO}_1^{\mathsf{D}} \Rightarrow 1] - \Pr[\text{REPRO}_0^{\mathsf{D}} \Rightarrow 1]| \leq \frac{3R}{2}\sqrt{qp_{\max}} \ ,$$

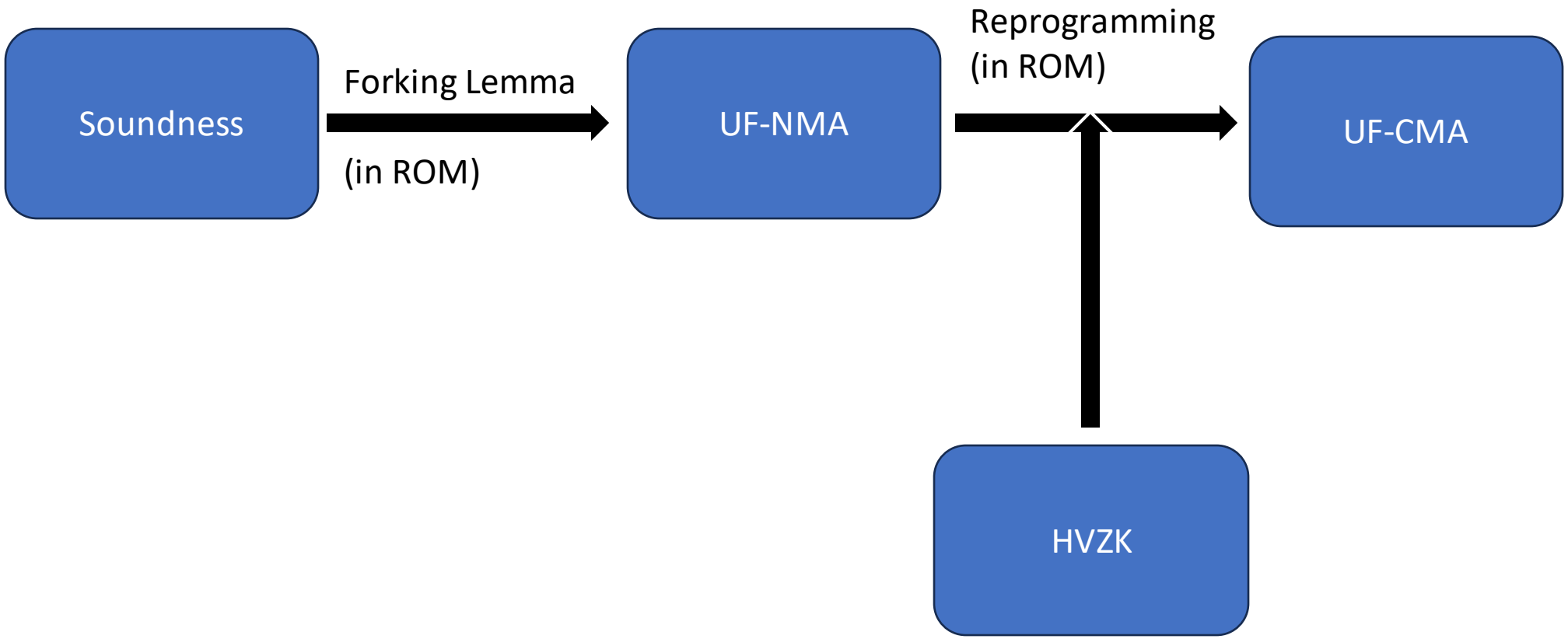
where $p_{\max} := \max_{x_1} p(x_1)$.

Proof intuition

- y is independtly, uniformly distributed.
- Reprogramming is perfectly indistinguishable if distinguisher D did not query H(x) before programming!
- Max probability of x is p_{max}.
- Pr[D queried H(x) before programming] = qp_{max}.

Back to the proof!

Proof outline



Game setup

Game_i(A)

1. (pk,sk) <- KeyGen()
2. (M*, σ*) <- $A^{\text{SignO}_i,H}$(pk)
3. Return (Verify(pk, M*, σ*) && && M* \notin L_M)

SignO_0(sk, M)

1. L_M U {M}
2. w <- P.Commit(sk)
3. c <- H(pk, w, m)
4. z <- P.Response(sk, w, c)
5. Return sig = (w, z)

Game hopping proof:

$A^{\text{SignO}_0,H}$(pk) ~ $A^{\text{SignO}_1,H}$(pk) ~ $A^{\text{SignO}_2,H}$(pk) ~ $A^{\text{SignO}_3,H}$(pk)

Hop 1

SignO_0(sk, M) [CMA-Sign]

1. L_M U {M}
2. w <- P.Commit(sk)
3. c <- H(pk, w, m)
4. z <- P.Response(sk, w, c)
5. Return sig = (w, z)

SignO_1(sk, M)

1. L_M U {M}
2. w <- P.Commit(sk)
3. c <-$_R$ CSpace
4. Set H(pk, w, m) -> c
5. z <- P.Response(sk, w, c)
6. Return sig = (w, z)

RO Reprogramming => $A^{\text{SignO}_0,H}(pk) \sim A^{\text{SignO}_1,H}(pk)$

Hop 2

$SignO_1(sk, M)$

1. L_M U {M}
2. w <- P.Commit(sk)
3. c <-$_R$ CSpace
4. Set H(pk, w, m) -> c
5. z <- P.Response(sk, w, c)
6. Return sig = (w, z)

$SignO_2(sk, M)$

1. L_M U {M}
2. (w,c,z) <- Trans(sk)
3. Set H(pk, w, m) -> c
4. Return sig = (w, z)

$A^{SignO_1,H}(pk) = A^{SignO_2,H}(pk)$

Hop 3

$SignO_2(sk, M)$

1. L_M U {M}
2. (w,c,z) <- Trans(sk)
3. Set H(pk, w, m) -> c
4. Return sig = (w, z)

$SignO_3(pk, M)$

1. L_M U {M}
2. (w,c,z) <- Sim(pk)
3. Set H(pk, w, m) -> c
4. Return sig = (w, z)

HVZK => $A^{SignO_2,H}(pk) \sim A^{SignO_3,H}(pk)$

M^A against NMA using A against CMA

M^A(pk)

1. (M*, σ*) <- $A^{SignO_3,H(pk,\ .\ )}$(pk)
2. Return (M*, σ*)

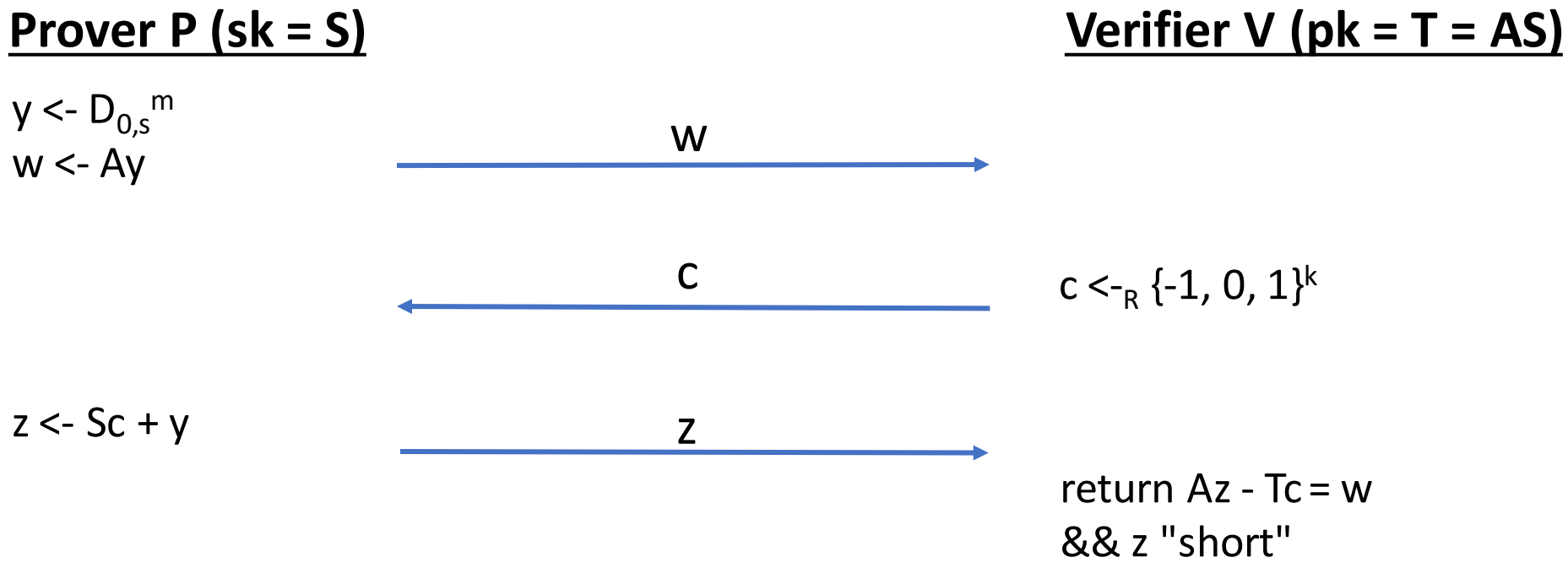
$SignO_3$(pk, M)

1. L_M U {M}
2. (w,c,z) <- Sim(pk)
3. Set H(pk, w, m) -> c
4. Return sig = (w, z)

$A^{SignO_0,H}$(pk) ~ $A^{SignO_3,H}$(pk) => UF-CMA(A) ~ UF-NMA(M^A)

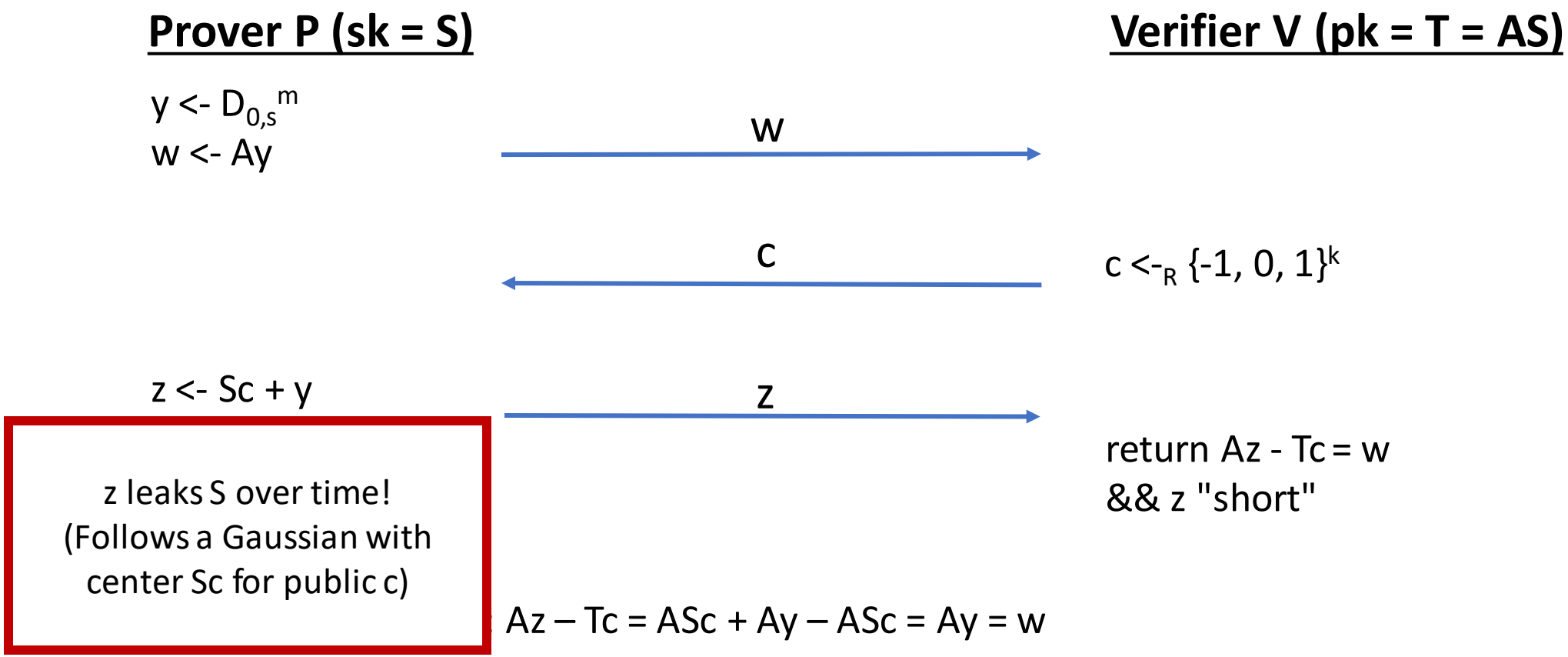
Fiat-Shamir with Aborts

Lattice-based Schnorr identification



Correctness: $Az - Tc = ASc + Ay - ASc = Ay = w$

Lattice-based Schnorr identification



FIXED Lattice-based Schnorr identification

Prover P (sk = S)

$y \leftarrow D_{0,s}^{m}$
$w \leftarrow Ay$

w →

c ←

$z \leftarrow Sc + y$
Accept z with probability
$\min(D_{0,s}^{m}(z) / MD_{Sc,s}^{m}(z), 1)$

z →

Verifier V (pk = T = AS)

$c \leftarrow_R \{-1, 0, 1\}^k$

return $Az - Tc = w$
&& z "short"

FIXED Lattice-based Schnorr identification

Prover P (sk = S)

$y \leftarrow D_{0,s}^m$
$w \leftarrow Ay$

w →

c ←

$c \leftarrow_R \{-1, 0, 1\}^k$

$z \leftarrow Sc + y$
Accept z with probability
$\min(D_{0,s}^m(z) / MD_{Sc,s}^m(z), 1)$

z →

Verifier V (pk = T = AS)

return $Az - Tc = w$
&& z "short"

Rejection sampling (output statistically close to $D_{0,s}^m$)

Security - HVZK (Soundness out of scope)

[Recall verify checks Az - Tc = w]

ZKSim(pk = T = AS):

1. $c \leftarrow_R \{-1, 0, 1\}^k$
2. $z \leftarrow_R D_{0,s}^m$
3. $w \leftarrow Az - Tc$
4. Output(w, c, z)

Trans(sk = S):

1. Repeat until accept:
   a. $y \leftarrow D_{0,s}^m$
   b. $w \leftarrow Ay$
   c. $c \leftarrow_R \{-1, 0, 1\}^k$
   d. $z \leftarrow Sc + y$
   e. Accept z with probability $\min(D_{0,s}^m(z) / MD_{Sc,s}^m(z), 1)$
2. Output (w, c, z)

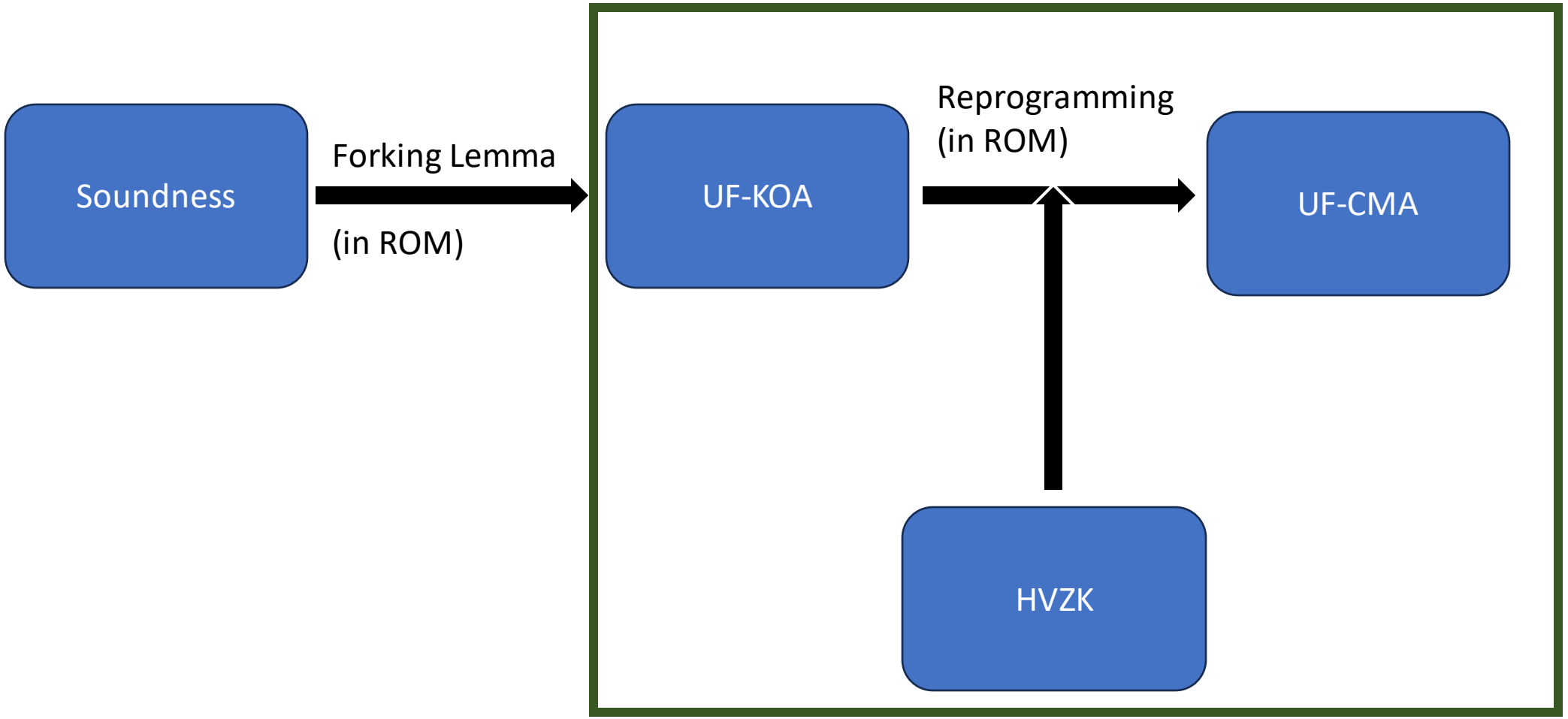
A subtle issue in the proof

Fixing and Mechanizing the Security Proof of Fiat-Shamir with Aborts and Dilithium. [pdf]

joint work with Manuel Barbosa, Gilles Barthe, Christian Doczkal, Jelle Don, Serge Fehr, Benjamin Grégoire, Yu-Hsuan Huang, Yi Lee, and Xiaodi Wu

CRYPTO 2023

Proof outline



The oracles (for an abstract proof)

$\mathsf{Sign}(m)$:

1: **repeat**
2: $(w, \mathsf{st}) \leftarrow \mathsf{Com}(sk)$
3: $c := H(w, m)$
4: $z := \mathsf{Resp}(w, c, \mathsf{st})$
5: **until** $z \neq \perp$
6: **return** (w, z)

$\mathsf{Prog}(m)$:

1: **repeat**
2: $(w, \mathsf{st}) \leftarrow \mathsf{Com}(sk)$
3: $H(w, m) := c \leftarrow C$
4: $z := \mathsf{Resp}(w, c, \mathsf{st})$
5: **until** $z \neq \perp$
6: **return** (w, z)

$\mathsf{Trans}(m)$:

1: **repeat**
2: $(w, \mathsf{st}) \leftarrow \mathsf{Com}(sk)$
3: $c \leftarrow C$
4: $z := \mathsf{Resp}(w, c, \mathsf{st})$
5: **until** $z \neq \perp$
6: $H(w, m) := c$
7: **return** (w, z)

$\mathsf{Sim}(m)$:

1: $(w, c, z) \leftarrow \mathsf{ZKSim}(pk)$
2: $H(w, m) := c$
3: **return** (w, z)

Fig. 2. Overview of the different oracles used for the hybrid proof.

The oracles (for an abstract proof)

Issue 1: Potentially unlimited # of iterations

Sign(m):

1: **repeat**
2: $(w, \mathsf{st}) \leftarrow \mathsf{Com}(sk)$
3: $c := H(w, m)$
4: $z := \mathsf{Resp}(w, c, \mathsf{st})$
5: **until** $z \neq \perp$
6: **return** (w, z)

Prog(m):

1: **repeat**
2: $(w, \mathsf{st}) \leftarrow \mathsf{Com}(sk)$
3: $H(w, m) := c \leftarrow C$
4: $z := \mathsf{Resp}(w, c, \mathsf{st})$
5: **until** $z \neq \perp$
6: **return** (w, z)

Trans(m):

1: **repeat**
2: $(w, \mathsf{st}) \leftarrow \mathsf{Com}(sk)$
3: $c \leftarrow C$
4: $z := \mathsf{Resp}(w, c, \mathsf{st})$
5: **until** $z \neq \perp$
6: $H(w, m) := c$
7: **return** (w, z)

Sim(m):

1: $(w, c, z) \leftarrow \mathsf{ZKSim}(pk)$
2: $H(w, m) := c$
3: **return** (w, z)

Fig. 2. Overview of the different oracles used for the hybrid proof.

The oracles (for an abstract proof)

Issue 1: Potentially unlimited # of iterations

$\mathsf{Sign}(m)$:

1: **repeat**
2: $(w, \mathsf{st}) \leftarrow \mathsf{Com}(sk)$
3: $c := H(w, m)$
4: $z := \mathsf{Resp}(w, c, \mathsf{st})$
5: **until** $z \neq \perp$
6: **return** (w, z)

$\mathsf{Prog}(m)$:

1: **repeat**
2: $(w, \mathsf{st}) \leftarrow \mathsf{Com}(sk)$
3: $H(w, m) := c \leftarrow C$
4: $z := \mathsf{Resp}(w, c, \mathsf{st})$
5: **until** $z \neq \perp$
6: **return** (w, z)

$\mathsf{Trans}(m)$:

1: **repeat**
2: $(w, \mathsf{st}) \leftarrow \mathsf{Com}(sk)$
3: $c \leftarrow C$
4: $z := \mathsf{Resp}(w, c, \mathsf{st})$
5: **until** $z \neq \perp$
6: $H(w, m) := c$
7: **return** (w, z)

$\mathsf{Sim}(m)$:

1: $(w, c, z) \leftarrow \mathsf{ZKSim}(pk)$
2: $H(w, m) := c$
3: **return** (w, z)

Issue 2: Trans only reprograms on "accepting (w,c)"

Fig. 2. Overview of the different oracles used for the hybrid proof.

Solving issue 1

- Fine-grained hybrid argument (one hop per hash call made by sign -> nested-hybrid over sign calls, and over hash calls in sign)
- Can bound distinguishing advantage between switching a full sign call considering that later hash calls are only reached with decreasing probability -> can neglect contributions of those hybrids to bound as the probability they are reached is vanishing.

The hybrids (for one sign call)

$\underline{\mathsf{Sign}(m)}$:

1: **repeat**
2: $\quad (w, \mathsf{st}) \leftarrow \mathsf{Com}(sk)$
3: $\quad c := H(w, m)$
4: $\quad z := \mathsf{Resp}(w, c, \mathsf{st})$
5: **until** $z \neq \bot$
6: **return** (w, z)

$\underline{\mathsf{Hyb}^k(m)}$:

1: $i := 0$
2: **repeat**
3: $\quad (w, \mathsf{st}) \leftarrow \mathsf{Com}(sk)$
4: $\quad$ **if** $i < k$ **then**
5: $\quad\quad H(w, m) := c \leftarrow C$
6: $\quad$ **else**
7: $\quad\quad c := H(w, m)$
8: $\quad z := \mathsf{Resp}(w, c, \mathsf{st})$
9: $\quad i := i + 1$
10: **until** $z \neq \bot$
11: **return** (w, z)

$\underline{\mathsf{Prog}(m)}$:

1: **repeat**
2: $\quad (w, \mathsf{st}) \leftarrow \mathsf{Com}(sk)$
3: $\quad H(w, m) := c \leftarrow C$
4: $\quad z := \mathsf{Resp}(w, c, \mathsf{st})$
5: **until** $z \neq \bot$
6: **return** (w, z)

Fig. 3. The oracles Sign (left) and Prog (right), and Hyb^k in-between (middle).

Solving issue 2

Break game hop into two steps:

1. Program all RO calls
2. Unprogram RO calls that are rejected

Intuition:

- Step 1 is undetectable if A did not query H(w,m) before
  (w only known afterwards & w has high entropy)
- Step 2 is undetectable if A does not query H(w,m) after for rejected w
  (rejected w not published & w has high entropy)

Results (see https://eprint.iacr.org/2023/246.pdf)

- ROM & QROM bound for Fiat-Shamir with aborts
- Mechanized ROM proof for Dilithium
- Detailed analysis of the impact on concrete parameters (no impact)

- Concurrent work:
  Julien Devevey, Pouria Fallahpour, Alain Passelègue, Damien Stehlé. A Detailed Analysis of Fiat-Shamir with Aborts. Crypto'23
  Observed issue 2 by manual inspection.

Conclusion

- The security proof of Dilithium, BLISS, Lyu12, and other FSwA signature schemes was subtly flawed
- We observed the flaw when machine-checking the proof, fixed it, finished machine-check, and ensured that there is no practical impact
- Did not touch on soundness in this talk
  - Dilithium: essentially rephrased as assumption;
  - [KLS'18]: with slightly different params, proof via lossy-IDS (dual-mode instance generator)
- PQ adds complexity which can introduce subtle flaws in proofs (even in without considering the QROM!)
- Issues went unnoticed for 10 years!

-> machine-checking these parts can guarantee absence of such issues.