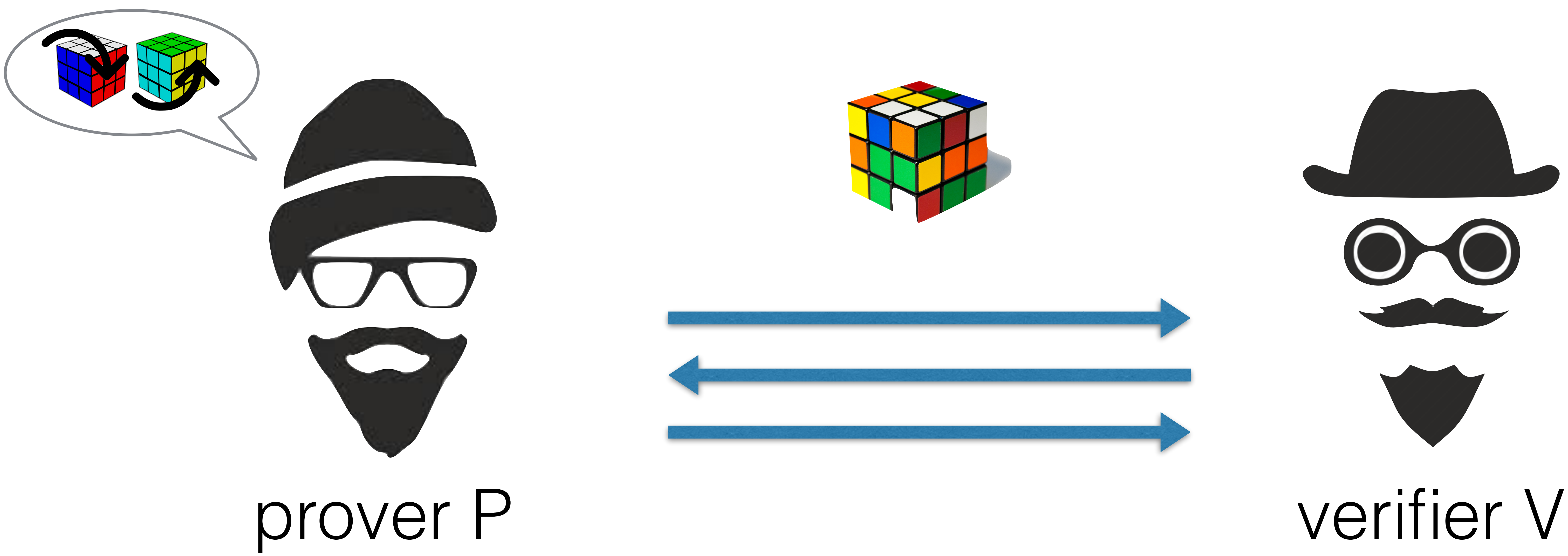
Towards Non-Interactive Zero-Knowledge Proofs from CDH and LWE

Geoffroy Couteau, Dennis Hofheinz

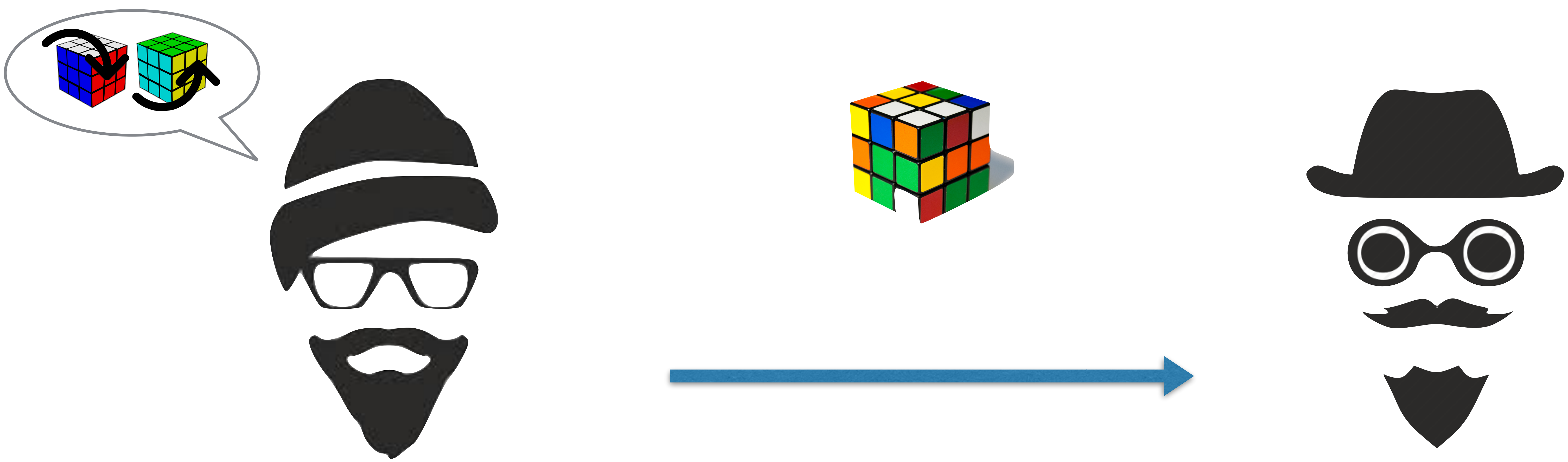
Karlsruher Institut für Technologie

Zero-Knowledge Proof



- Complete: if P knows a solution, V accepts
- Sound: if there is no solution, P cannot convince V
- Zero-Knowledge: V does not learn the solution

Non-Interactive Zero-Knowledge Proof



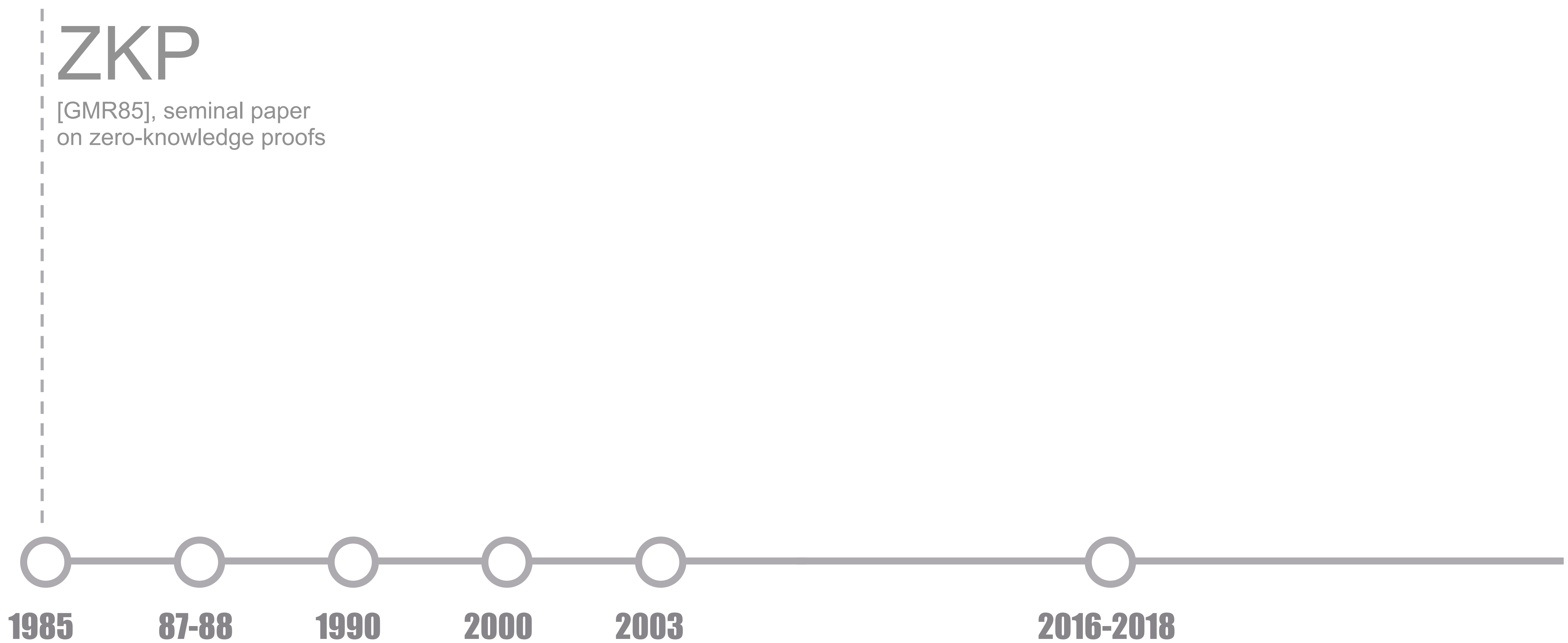
prover P

verifier V

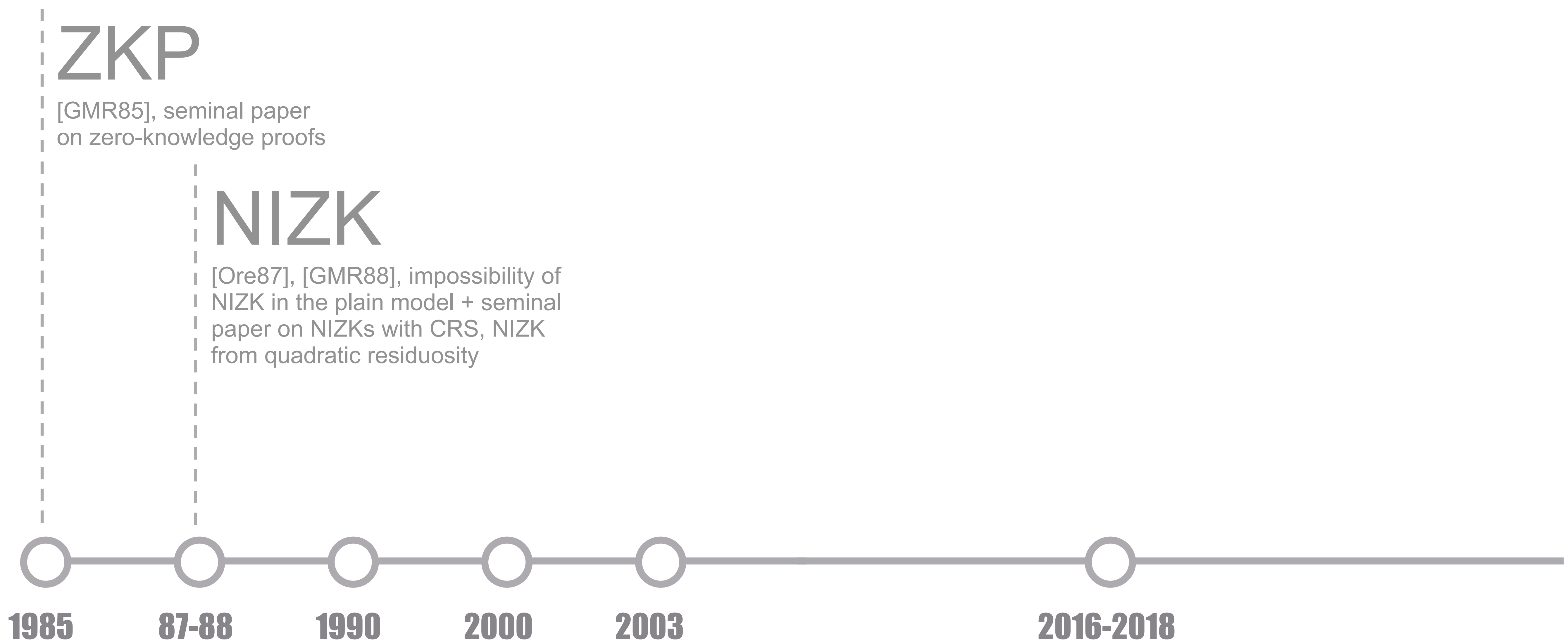
- Complete: if P knows a solution, V accepts
- Sound: if there is no solution, P cannot convince V
- Zero-Knowledge: V does not learn the solution

Brief History of NIZKs

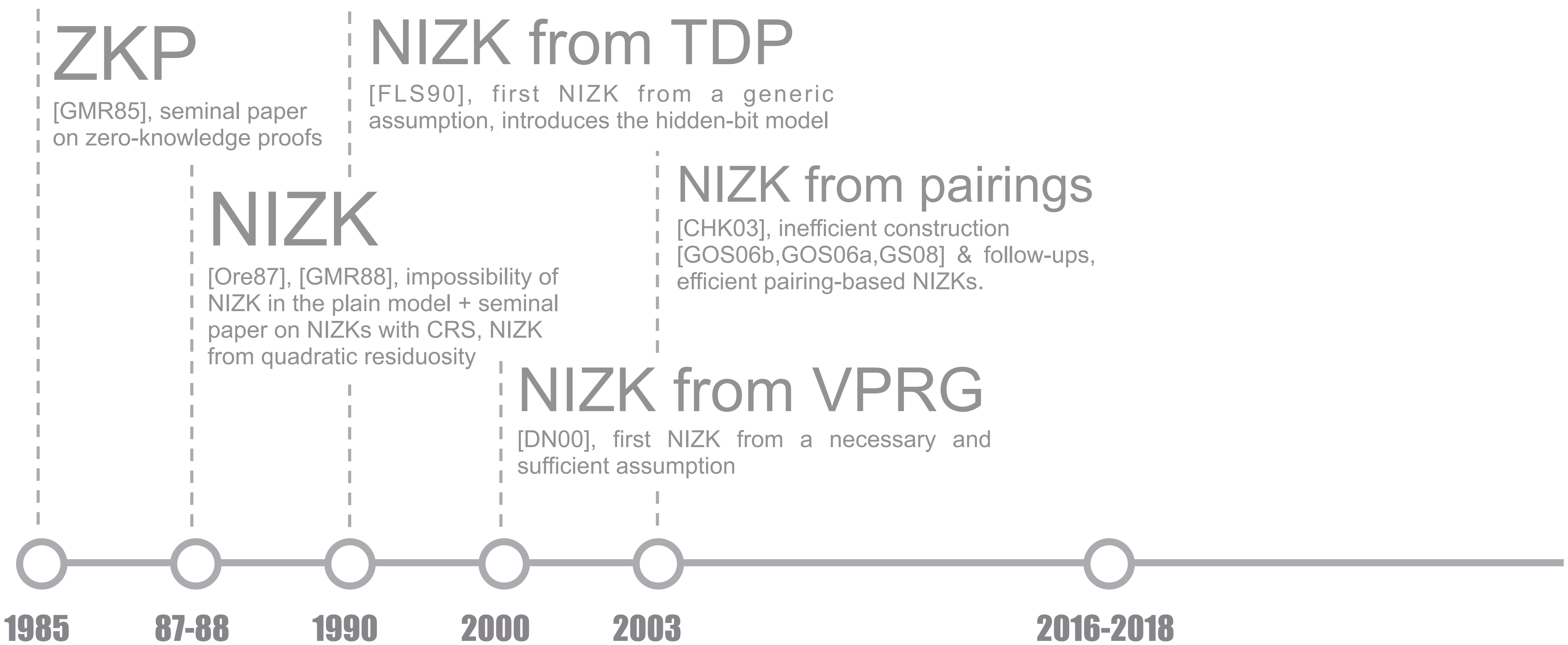
Brief History of NIZKs



Brief History of NIZKs



Brief History of NIZKs



Brief History of NIZKs

1985

ZKP

[GMR85], seminal paper on zero-knowledge proofs

87-88

NIZK

[Ore87], [GMR88], impossibility of NIZK in the plain model + seminal paper on NIZKs with CRS, NIZK from quadratic residuosity

1990

NIZK from TDP

[FLS90], first NIZK from a generic assumption, introduces the hidden-bit model

2000

NIZK from VPRG

[DN00], first NIZK from a necessary and sufficient assumption

2003

NIZK from pairings

[CHK03], inefficient construction [GOS06b,GOS06a,GS08] & follow-ups, efficient pairing-based NIZKs.

2016-2018

NIZK from new assumptions

[CCR16], [KRR17], [CCRR18], [HL18], [CCH+18], [CLW18]: instantiating correlation intractable hash functions (iO, exponentially-strong KDM security, circular FHE)

[RR18]: NIZK from LWE + NIZK for BDD

Brief History of NIZKs

1985 — ZKP

[GMR85], seminal paper on zero-knowledge proofs

87-88 — NIZK

[Ore87], [GMR88], impossibility of NIZK in the plain model + seminal paper on NIZKs with CRS, NIZK from quadratic residuosity

1990 — NIZK from TDP

[FLS90], first NIZK from a generic assumption, introduces the hidden-bit model

2000 — NIZK from VPRG

[DN00], first NIZK from a necessary and sufficient assumption

2003 — NIZK from pairings

[CHK03], inefficient construction [GOS06b,GOS06a,GS08] & follow-ups, efficient pairing-based NIZKs.

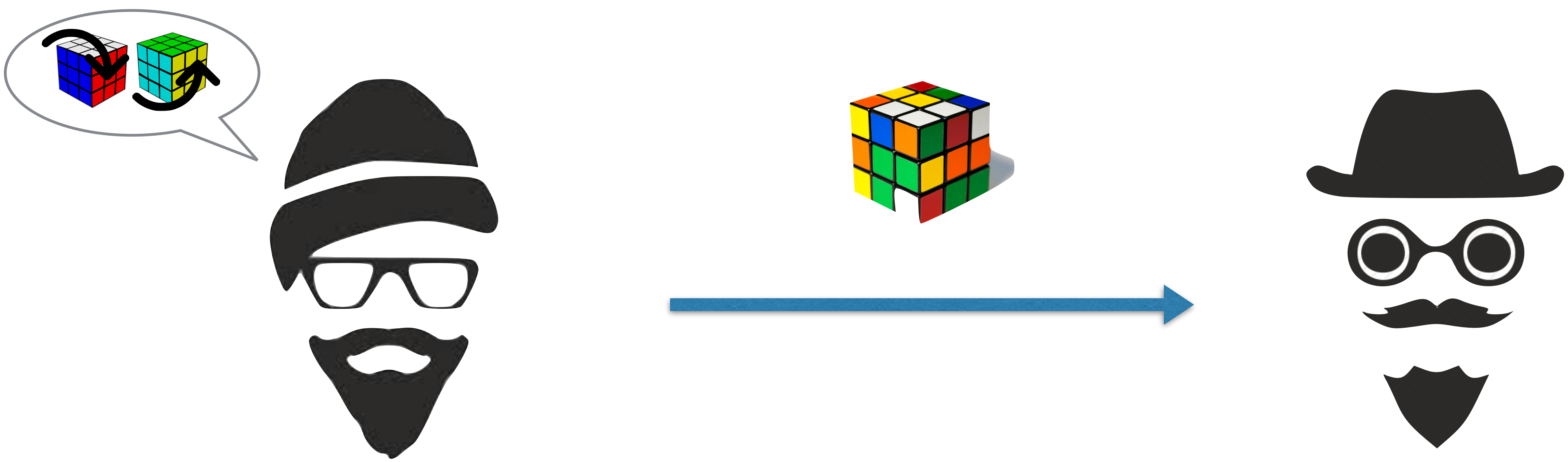
2016-2018 — NIZK from new assumptions

[CCR16], [KRR17], [CCRR18], [HL18], [CCH+18], [CLW18]: instantiating correlation intractable hash functions (iO, exponentially-strong KDM security, circular FHE)

[RR18]: NIZK from LWE + NIZK for BDD

2003 – 2016-2018: Investigating relaxed notions

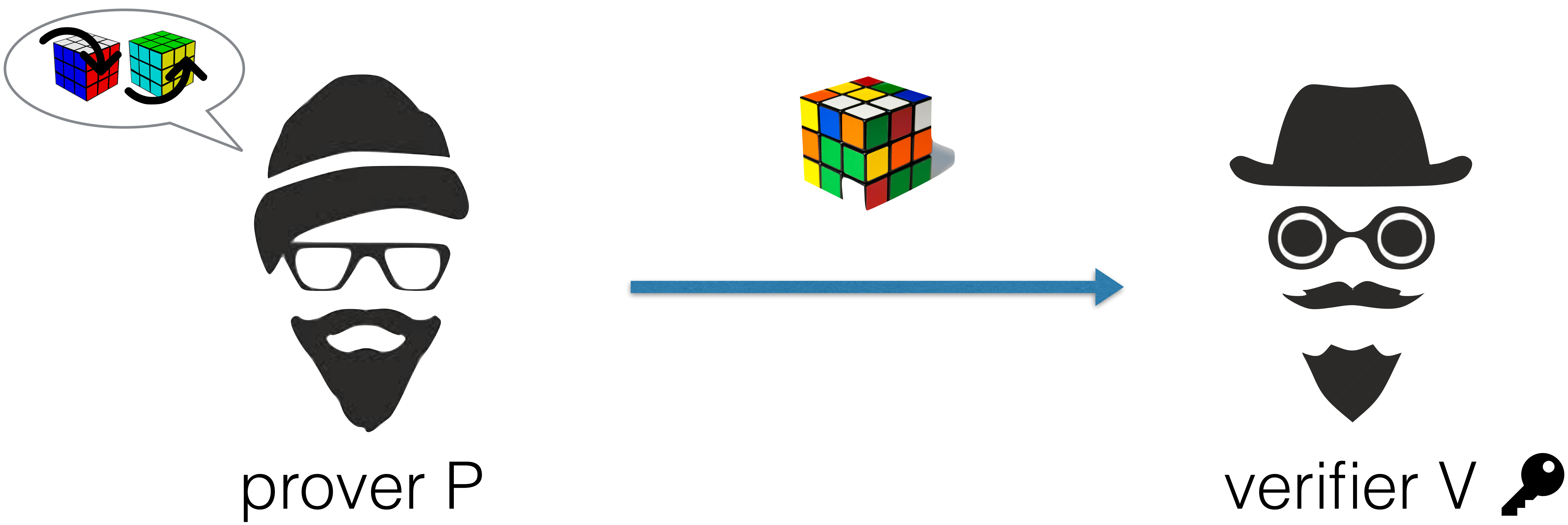
Designated-Verifier NIZK



prover P verifier V

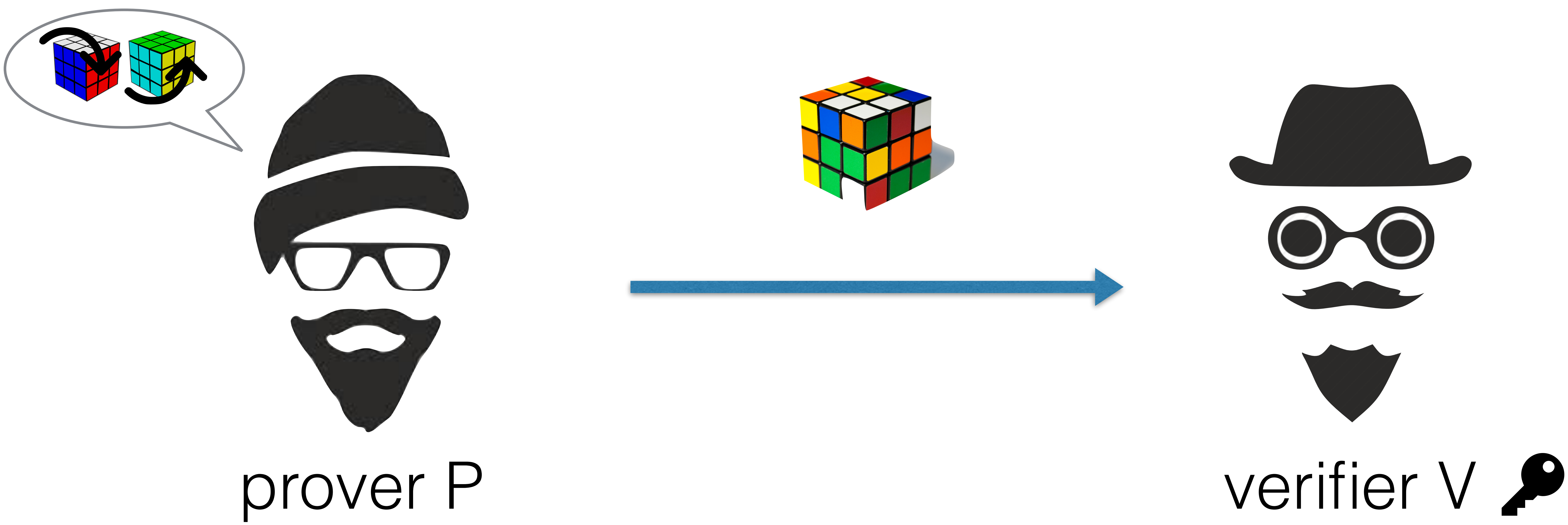
- Complete: if P knows a solution, V accepts
- Sound: if there is no solution, P cannot convince V
- Zero-Knowledge: V does not learn the solution

Designated-Verifier NIZK



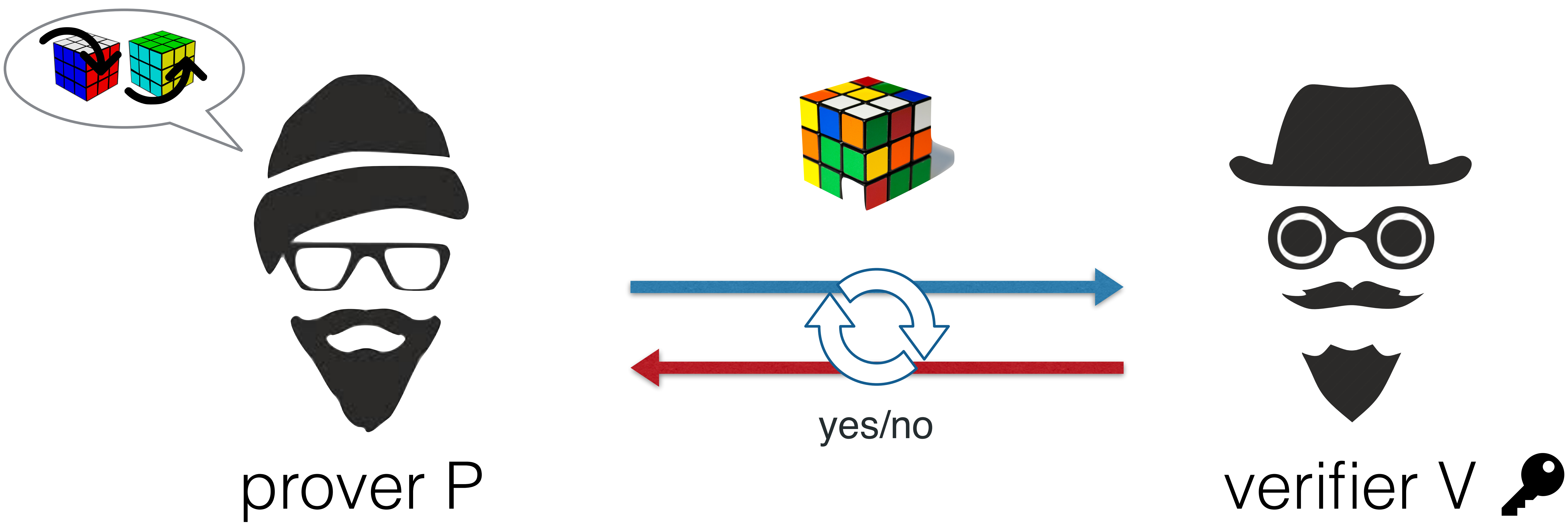
- Complete: if P knows a solution, V accepts
- Sound: if there is no solution, P cannot convince V
- Zero-Knowledge: V does not learn the solution

Designated-Verifier NIZK



- Complete: if P knows a solution, V accepts
- Unbounded Sound: if there is no solution, P cannot convince V
- Zero-Knowledge: V does not learn the solution

Designated-Verifier NIZK



- Complete: if P knows a solution, V accepts
- Unbounded Sound: if there is no solution, P cannot convince V
- Zero-Knowledge: V does not learn the solution

Brief History of (DV)NIZKs

1985 — ZKP
[GMR85], seminal paper on zero-knowledge proofs

87-88 — NIZK
[Ore87], [GMR88], impossibility of NIZK in the plain model + seminal paper on NIZKs with CRS, NIZK from quadratic residuosity

1990 — NIZK from TDP
[FLS90], first NIZK from a generic assumption, introduces the hidden-bit model

2000 — NIZK from VPRG
[DN00], first NIZK from a necessary and sufficient assumption

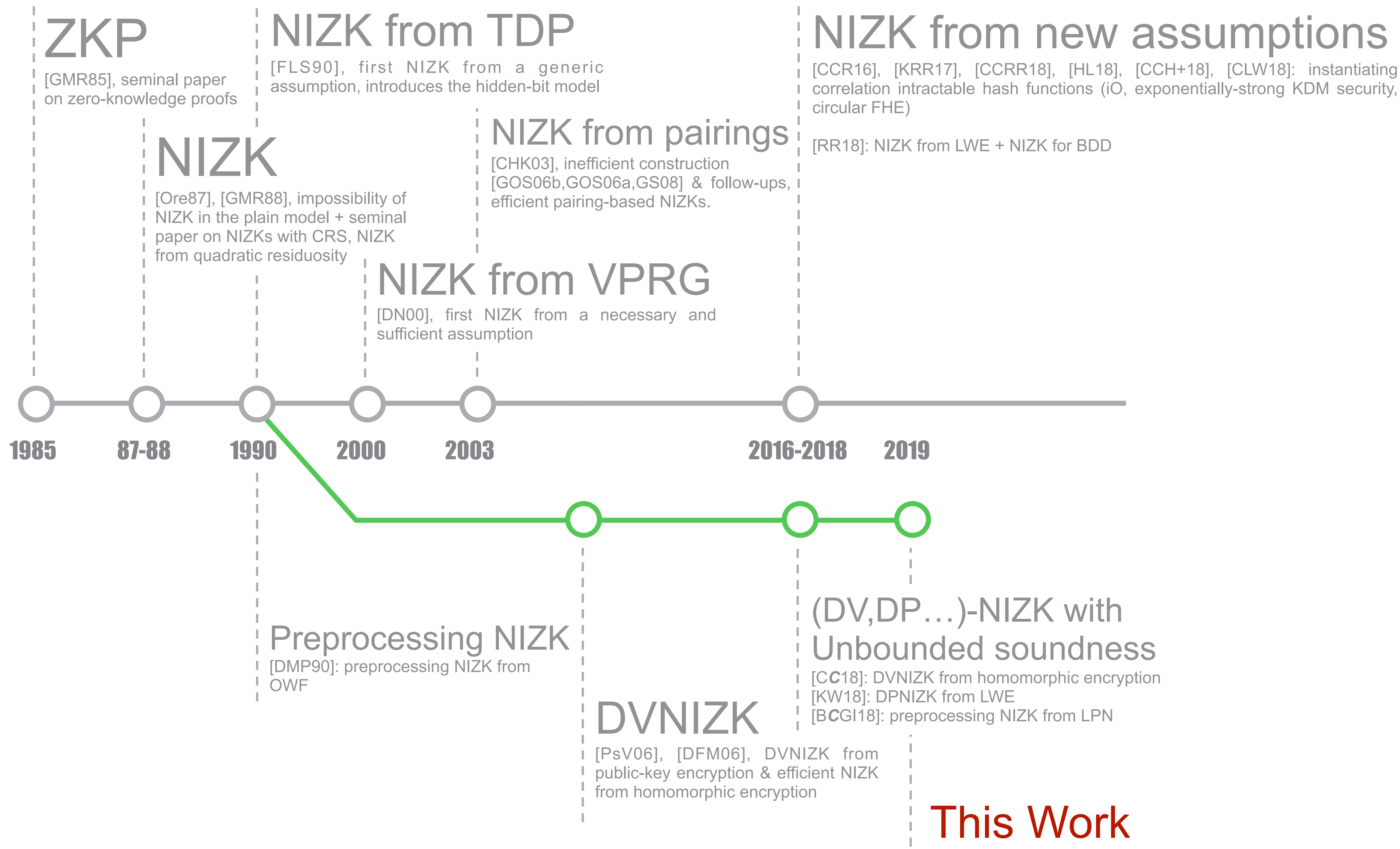
2003 — NIZK from pairings
[CHK03], inefficient construction
[GOS06b,GOS06a,GS08] & follow-ups, efficient pairing-based NIZKs.

2016-2018 — NIZK from new assumptions
[CCR16], [KRR17], [CCRR18], [HL18], [CCH+18], [CLW18]: instantiating correlation intractable hash functions (iO, exponentially-strong KDM security, circular FHE)

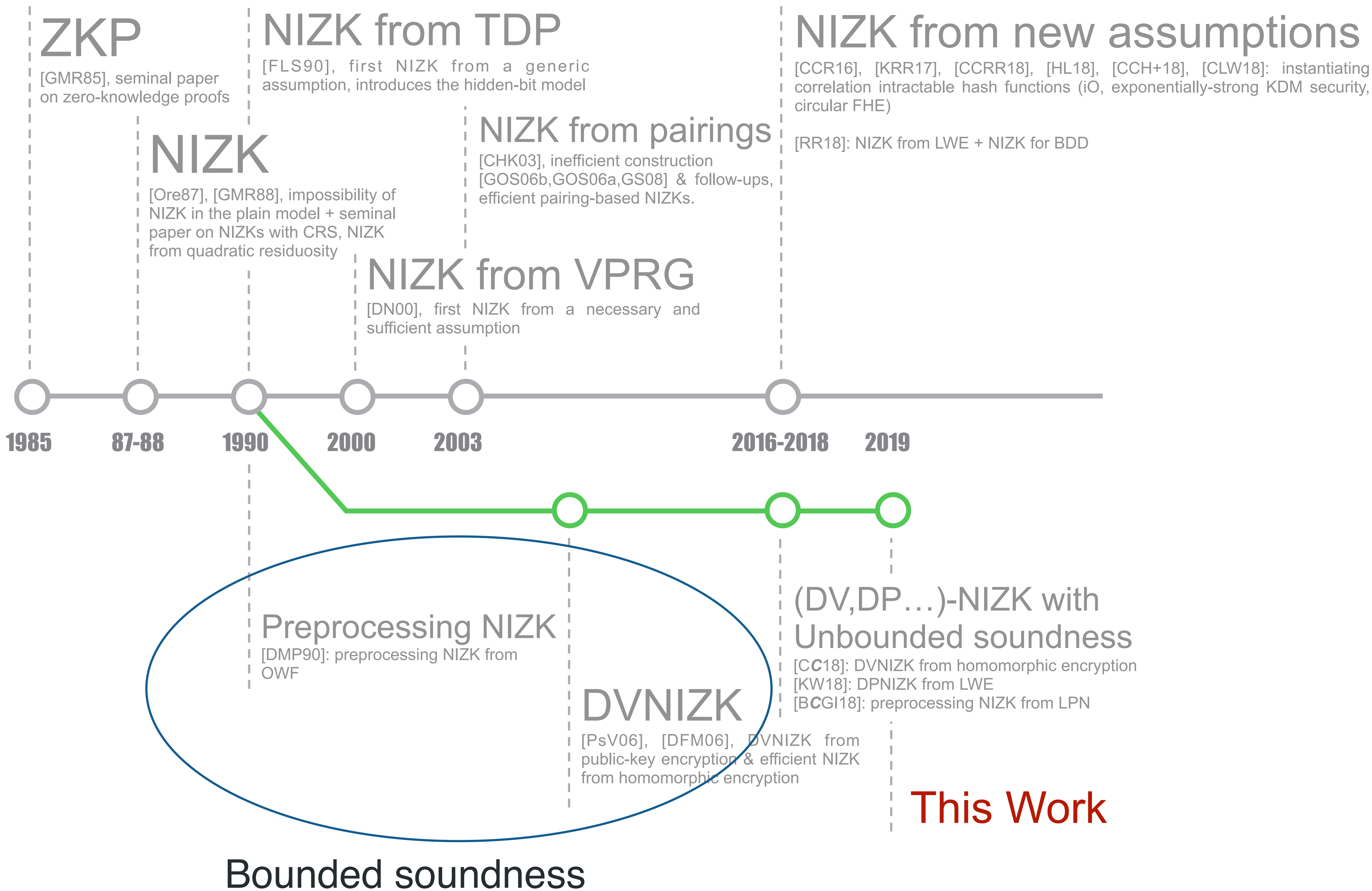
[RR18]: NIZK from LWE + NIZK for BDD

2019

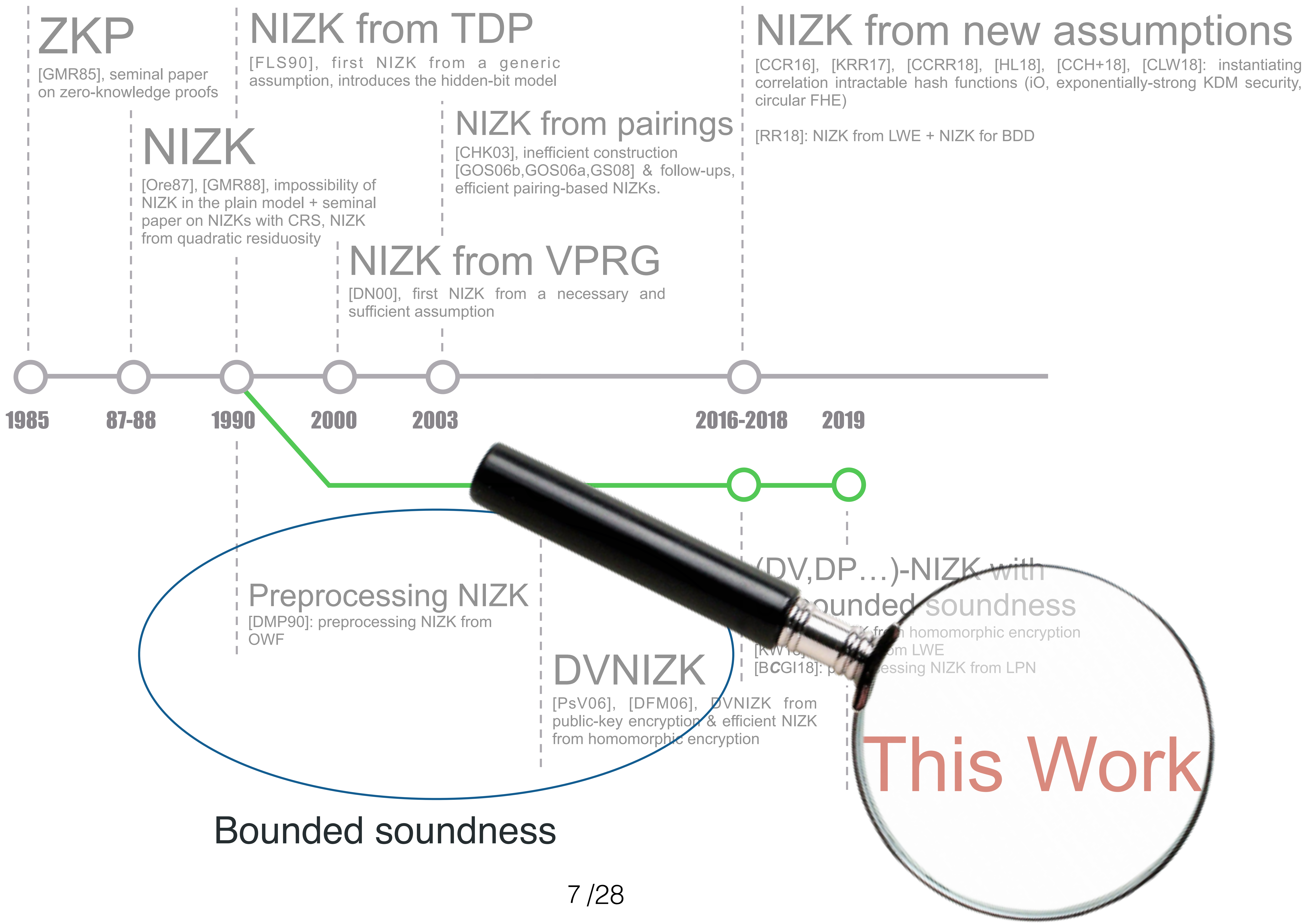
Brief History of (DV)NIZKs



Brief History of (DV)NIZKs



Brief History of (DV)NIZKs



Our Contribution

We obtain two new constructions:

1) A DVNIZK for NP under the CDH assumption

First direct indication that DVNIZK with unbounded soundness are actually easier to build than standard NIZK

2) A (DV)NIZK for NP assuming LWE and the existence of a (DV)NIWI for BDD

Improving over, and considerably simplifying, the recent result of [RR18] which required a NIZK for BDD.

Roadmap

[DN00]: $\boxed{\text{Verifiable Pseudorandom Generator}}$ + $\boxed{\text{NIZK in the hidden-bit model}}$ $\Rightarrow$ NIZK

Roadmap

[DN00]: Verifiable Pseudorandom Generator + NIZK in the hidden-bit model $\Rightarrow$ NIZK

$\Downarrow$

Verifiable Pseudorandom Generator:
- Relaxed soundness
- Generalization to the DV setting

Roadmap

[DN00]: Verifiable Pseudorandom Generator + NIZK in the hidden-bit model ⇒ NIZK

⇓

Verifiable Pseudorandom Generator:
- Relaxed soundness
- Generalization to the DV setting

+ NIZK in the hidden-bit model ⇒ NIZK

Roadmap

[DN00]: Verifiable Pseudorandom Generator + NIZK in the hidden-bit model ⇒ NIZK

⇓

Verifiable Pseudorandom Generator:
- Relaxed soundness
- Generalization to the DV setting

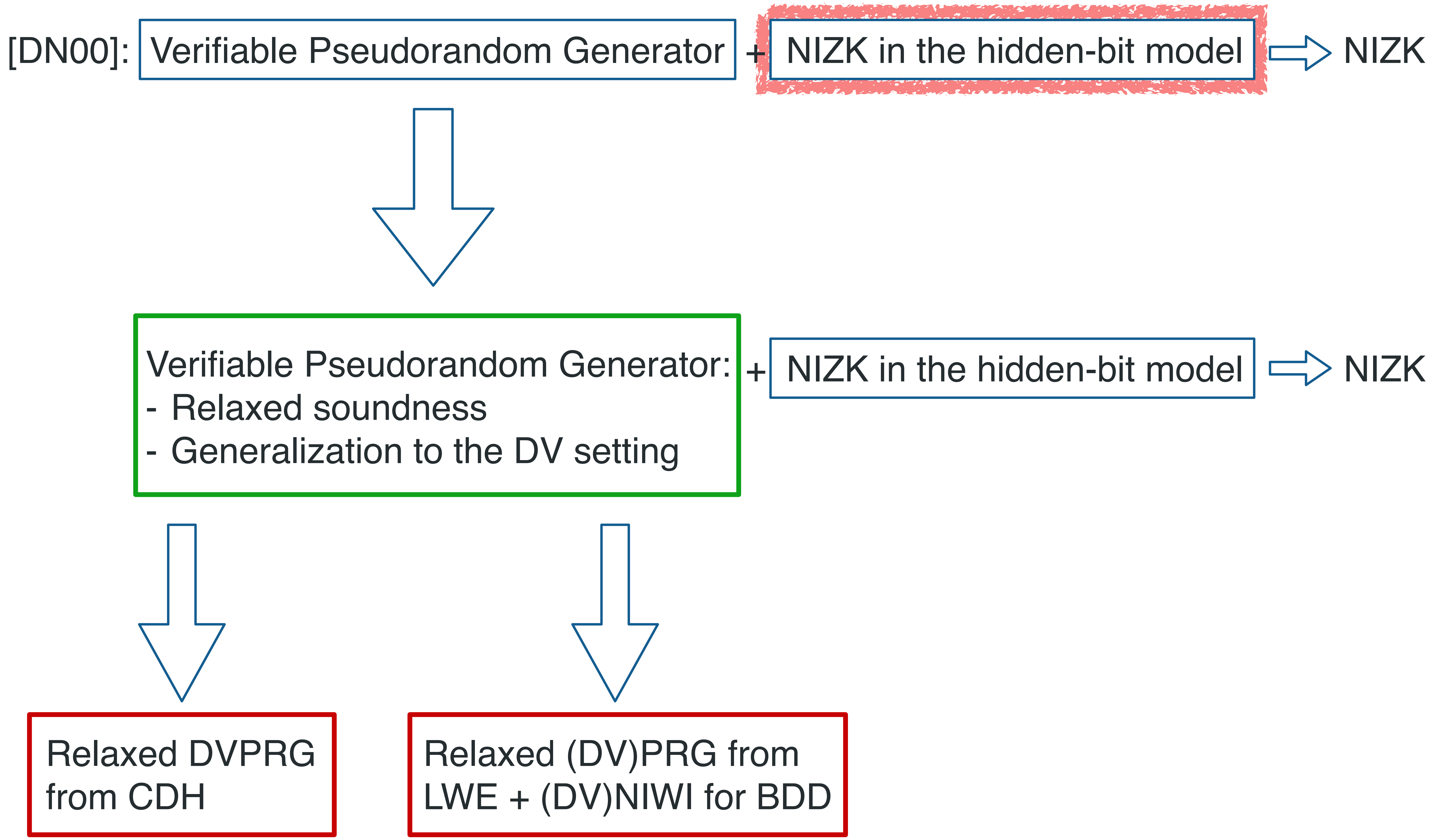
\+ NIZK in the hidden-bit model ⇒ NIZK

⇓ ⇓

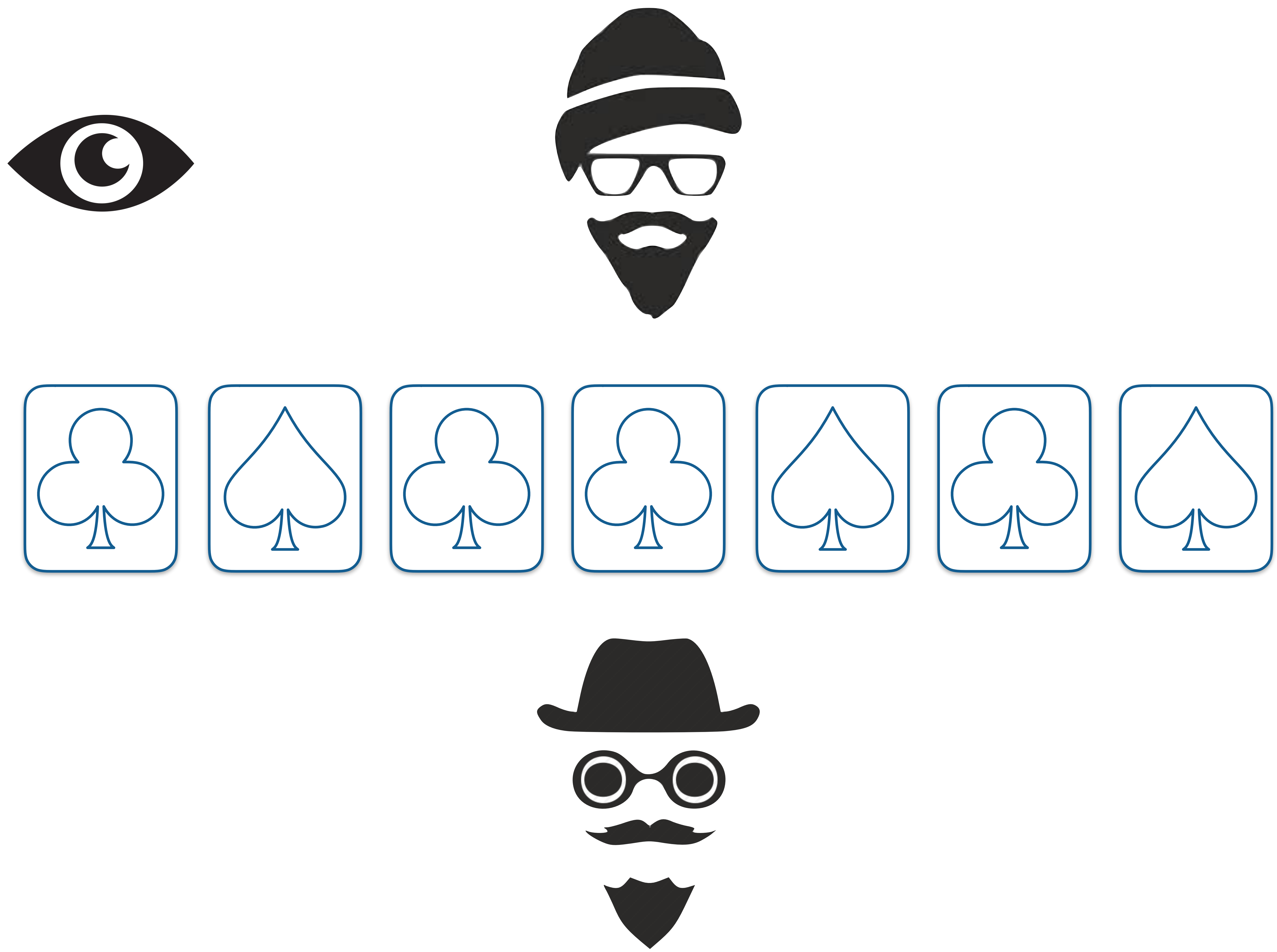
Relaxed DVPRG from CDH

Relaxed (DV)PRG from LWE + (DV)NIWI for BDD

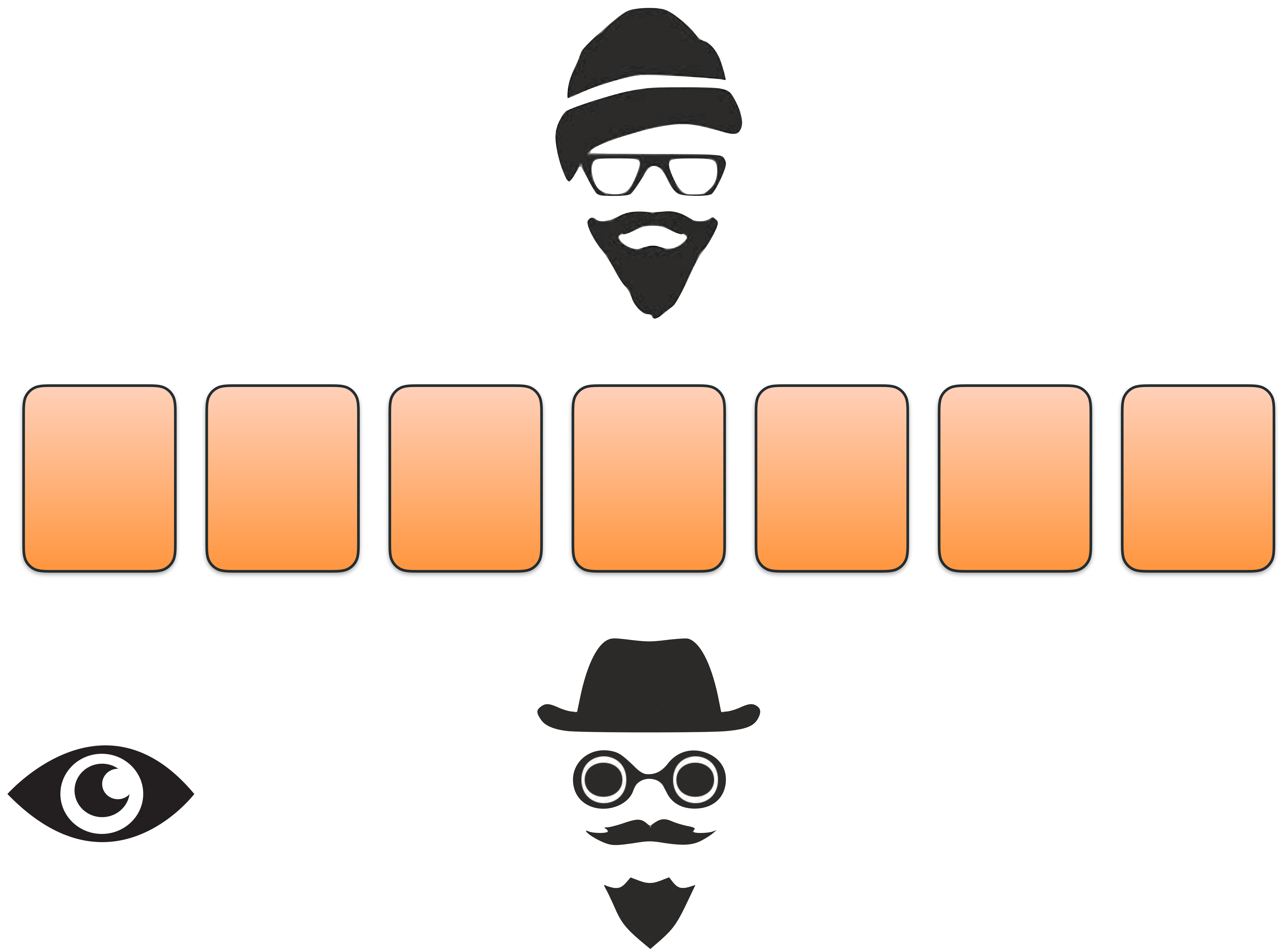
Roadmap


[DN00]:
Verifiable Pseudorandom Generator
+
NIZK in the hidden-bit model
NIZK
Verifiable Pseudorandom Generator:
- Relaxed soundness
- Generalization to the DV setting
+
NIZK in the hidden-bit model
NIZK
Relaxed DVPRG
from CDH
Relaxed (DV)PRG from
LWE + (DV)NIWI for BDD

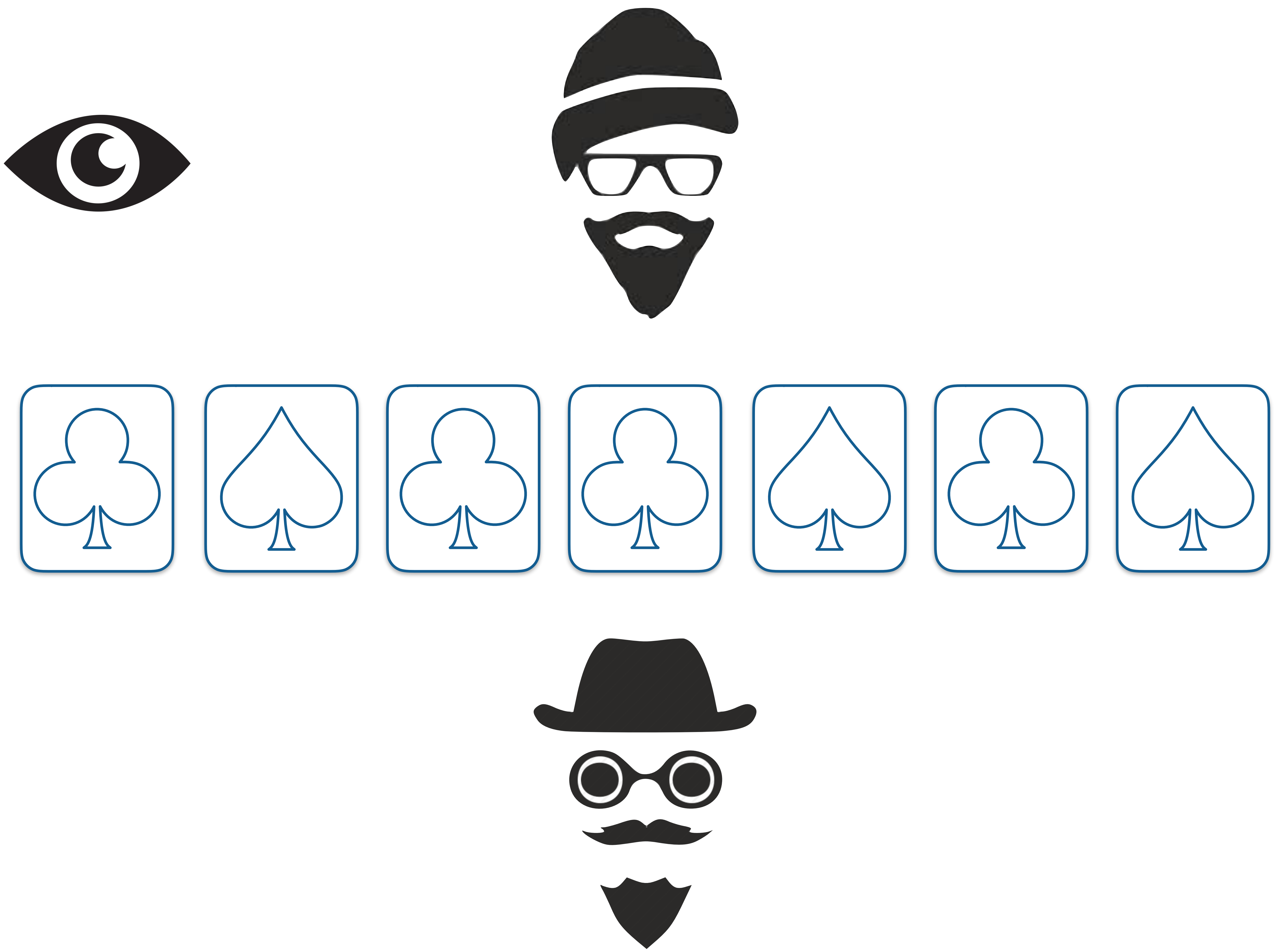
The Hidden-Bit Model



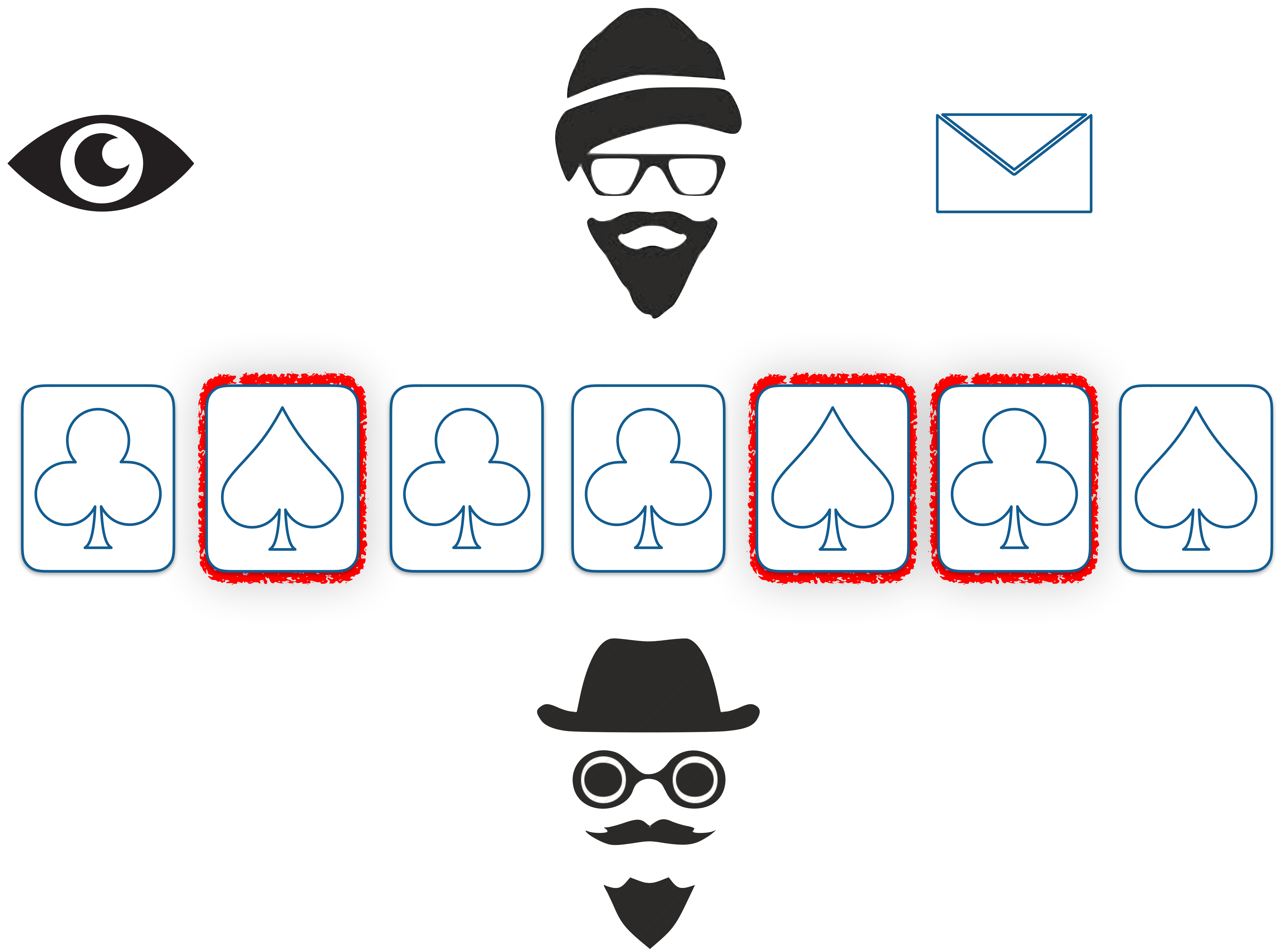
The Hidden-Bit Model



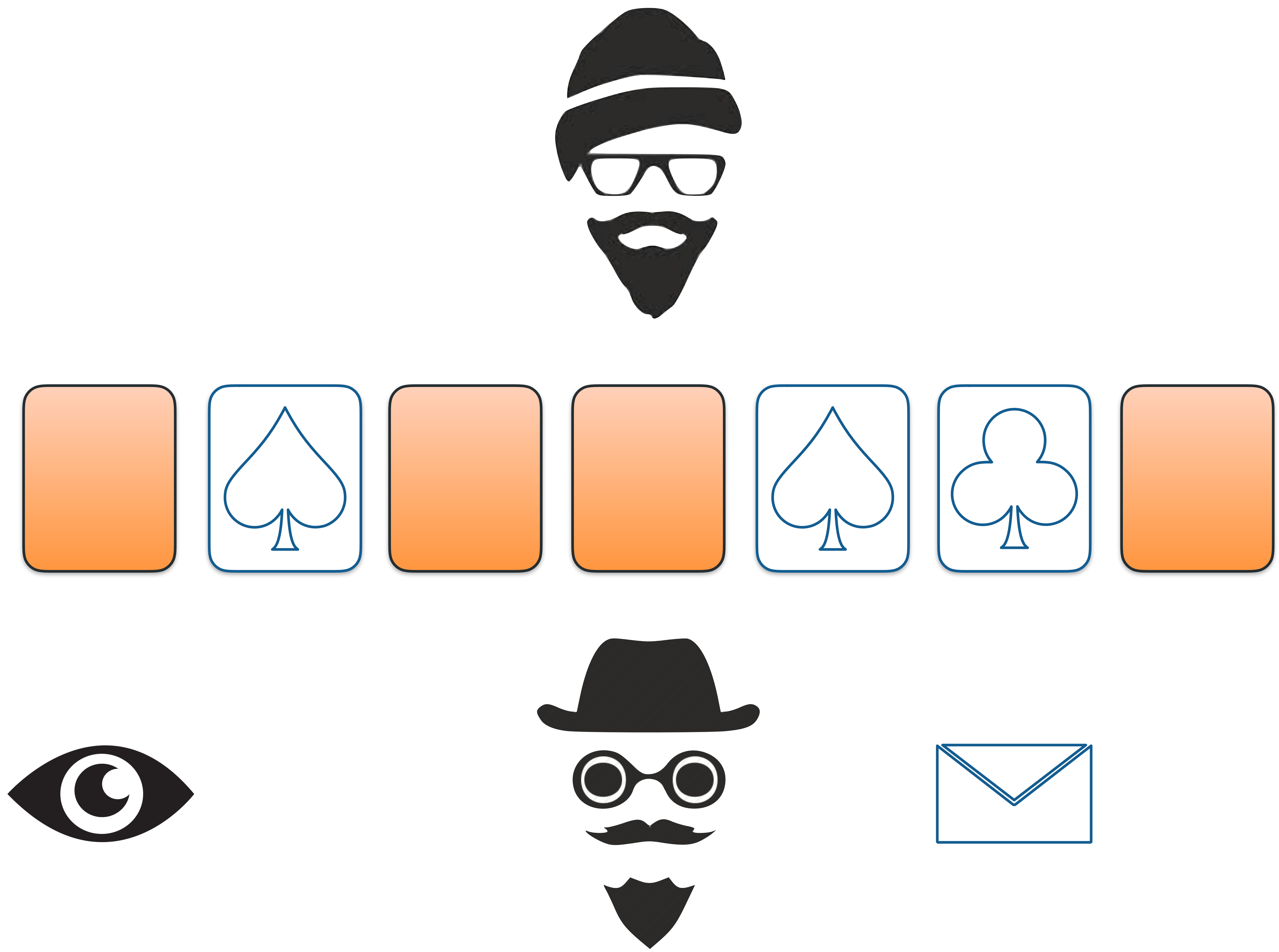
The Hidden-Bit Model



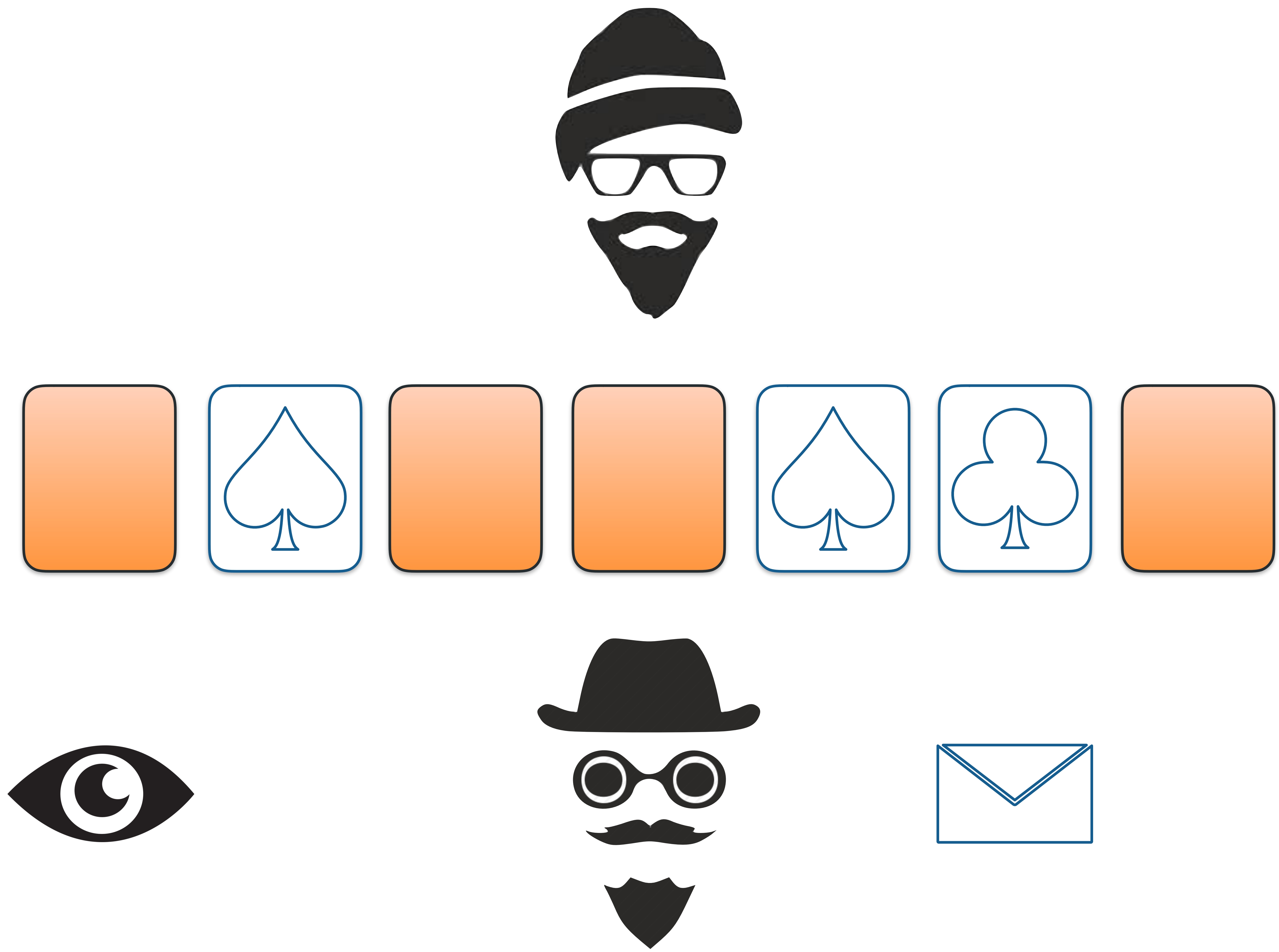
The Hidden-Bit Model



The Hidden-Bit Model



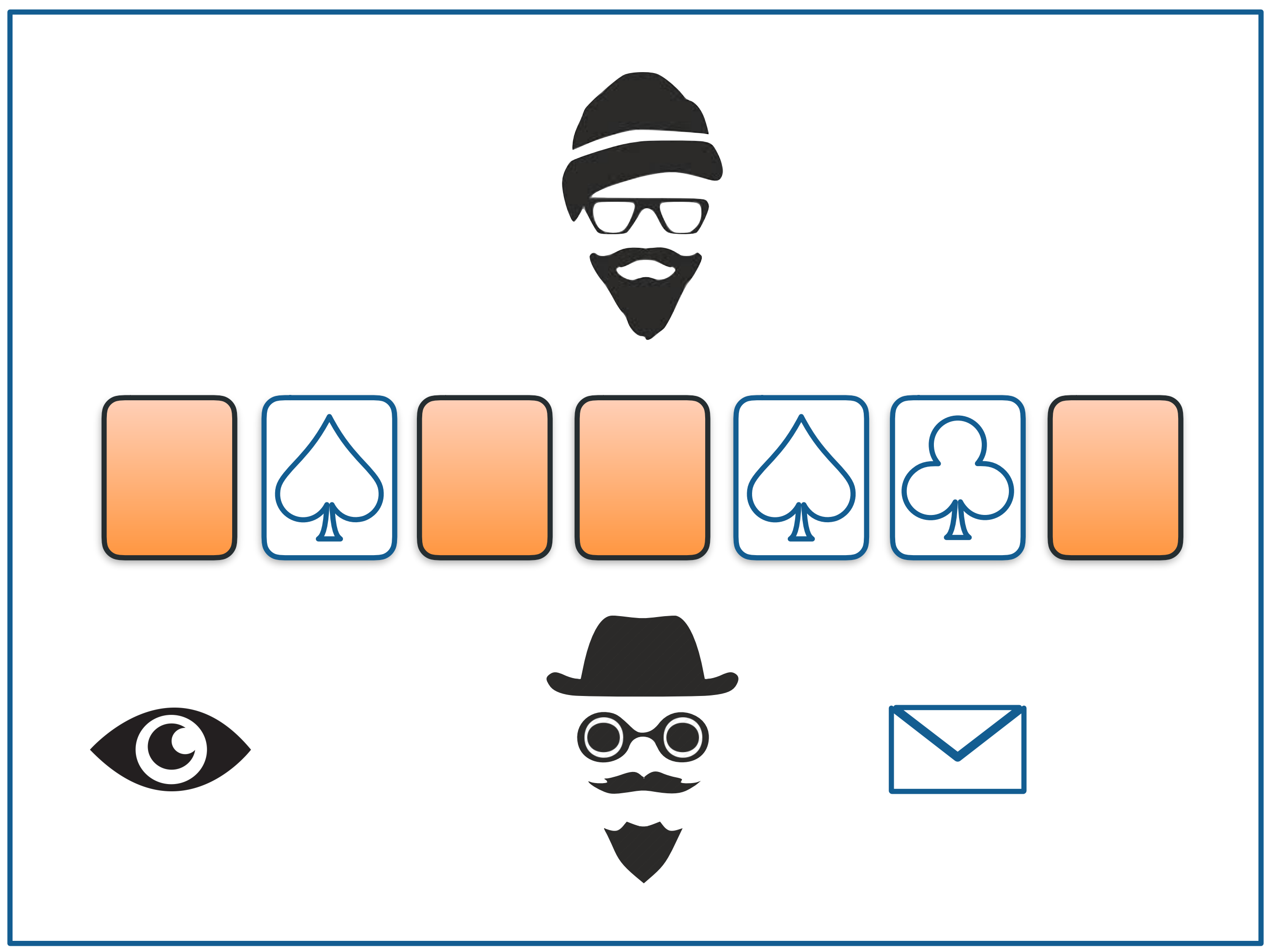
The Hidden-Bit Model

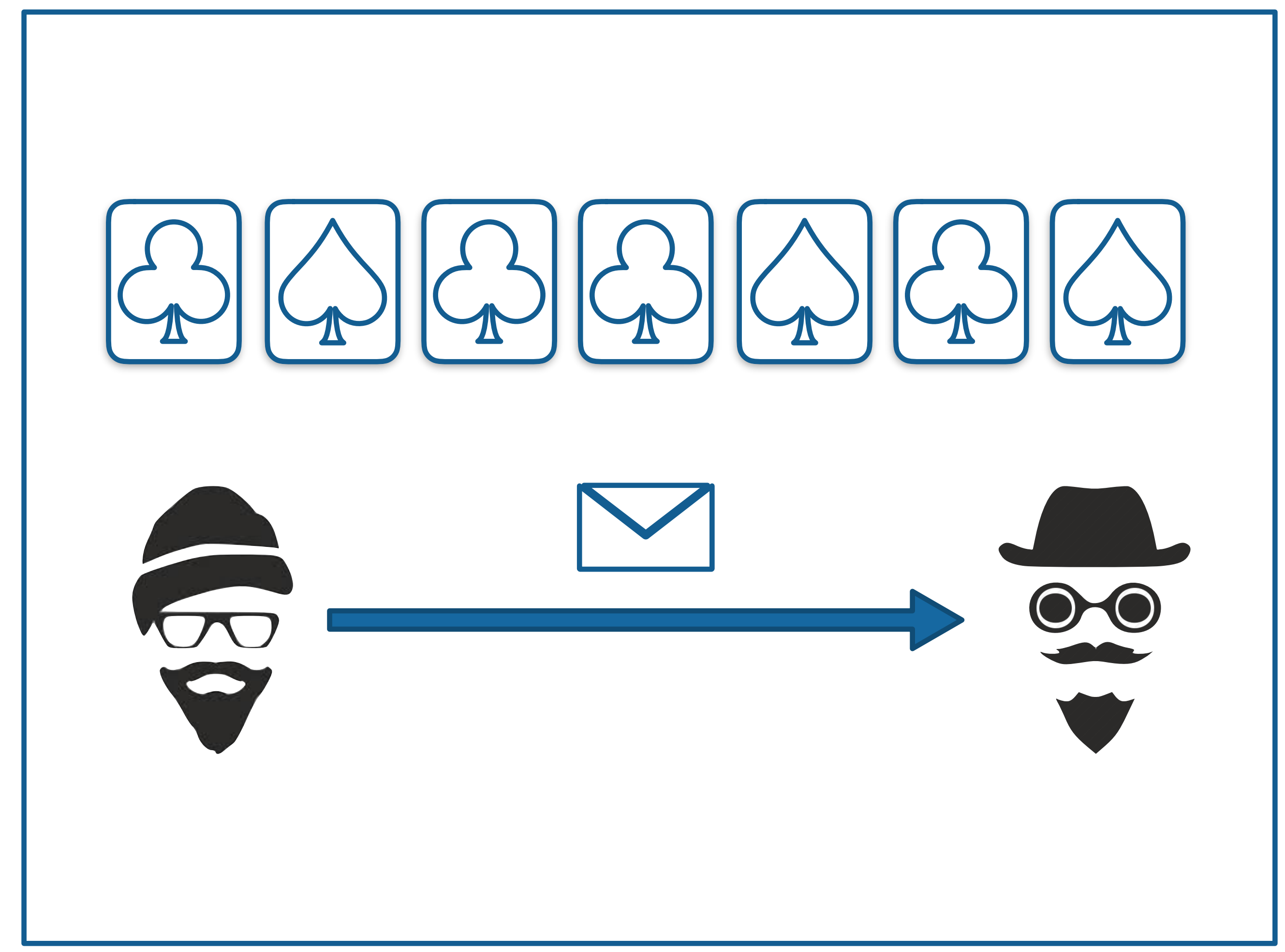


[FLS90]: NIZKs for NP exist unconditionally in the HBM

Instantiating The Hidden-Bit Model

Cryptographic primitive

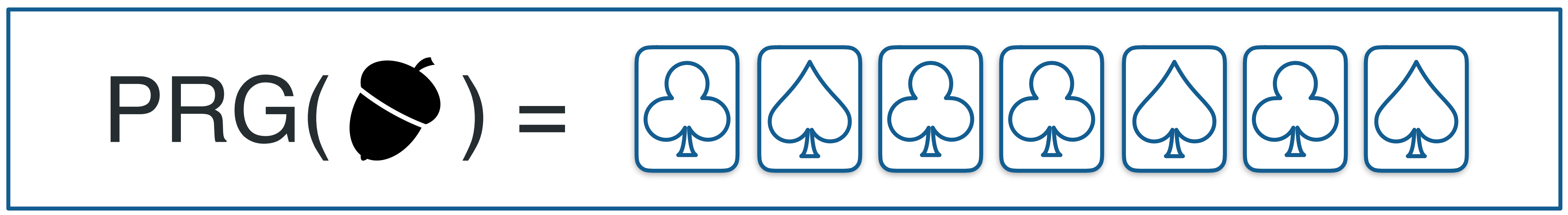


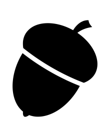
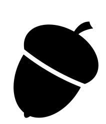


Prover's task, given the CRS:

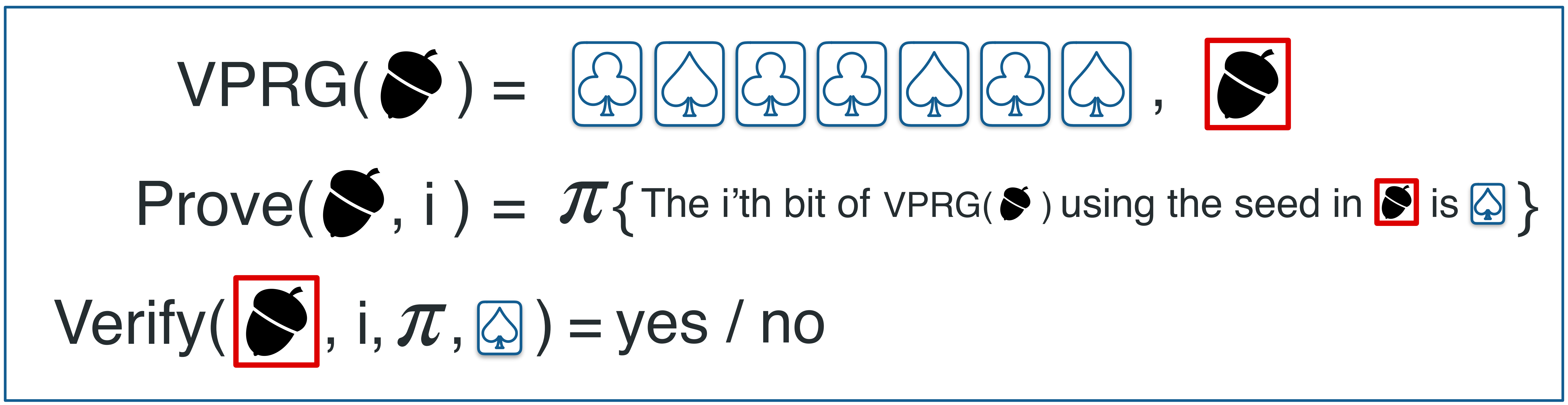
1. Produce a string which is indistinguishable from random
2. Be able to provably 'open' positions of this pseudorandom string
3. The openings should not reveal the non-opened positions

Pseudorandom Generators

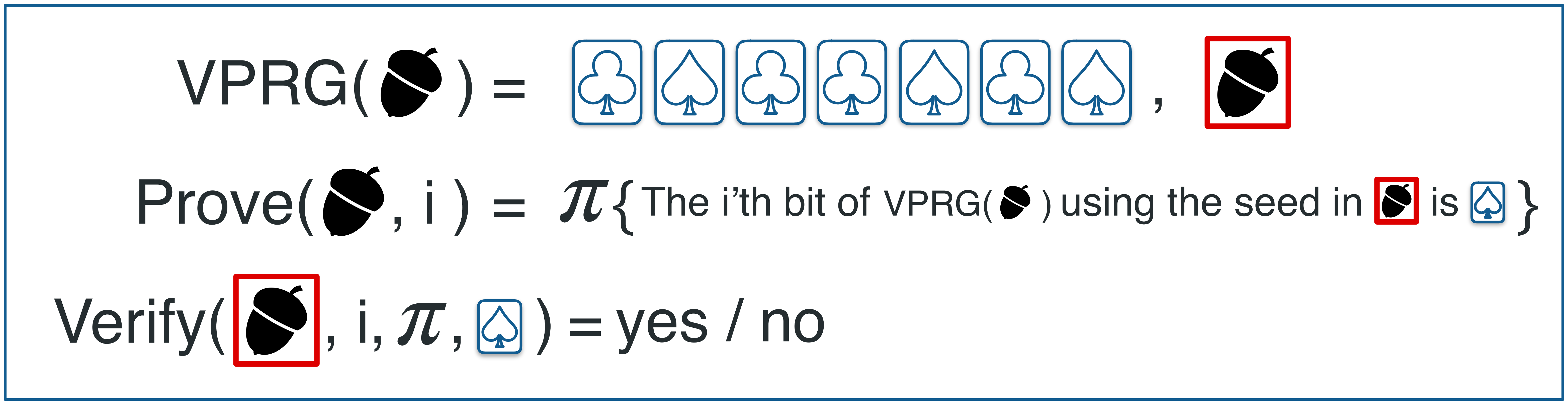


-  is short
- If  is random,  cannot be distinguished from a truly random string

Verifiable Pseudorandom Generators

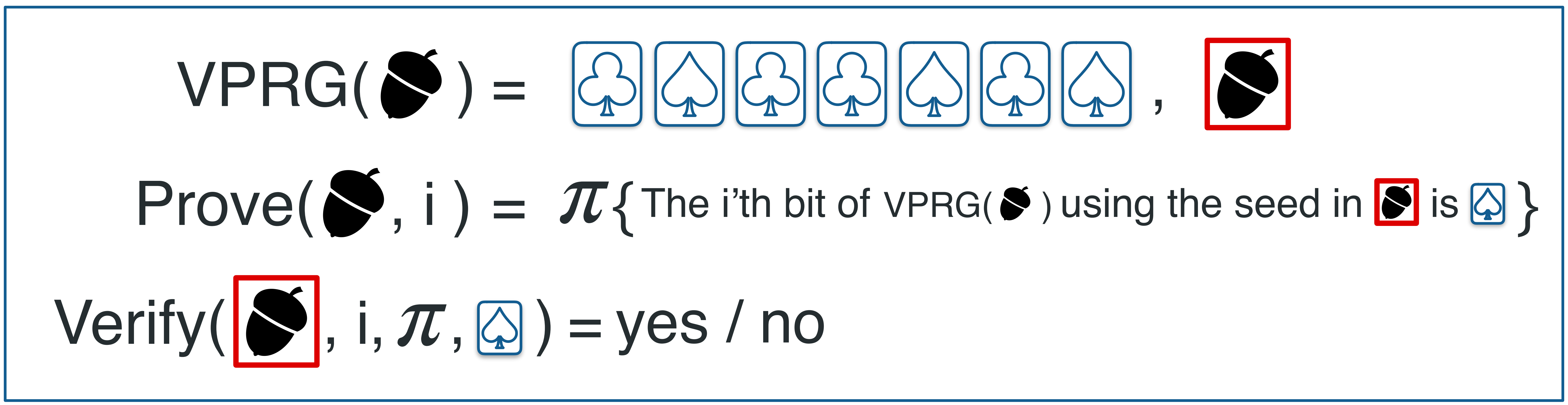


Verifiable Pseudorandom Generators



- Is short
- The proof leaks nothing more about
- The proof is sound in a strong sense

Verifiable Pseudorandom Generators



- Is short
- The proof leaks nothing more about
- **The proof is sound in a strong sense**

Verifiable Pseudorandom Generators

VPRG( ) = ,

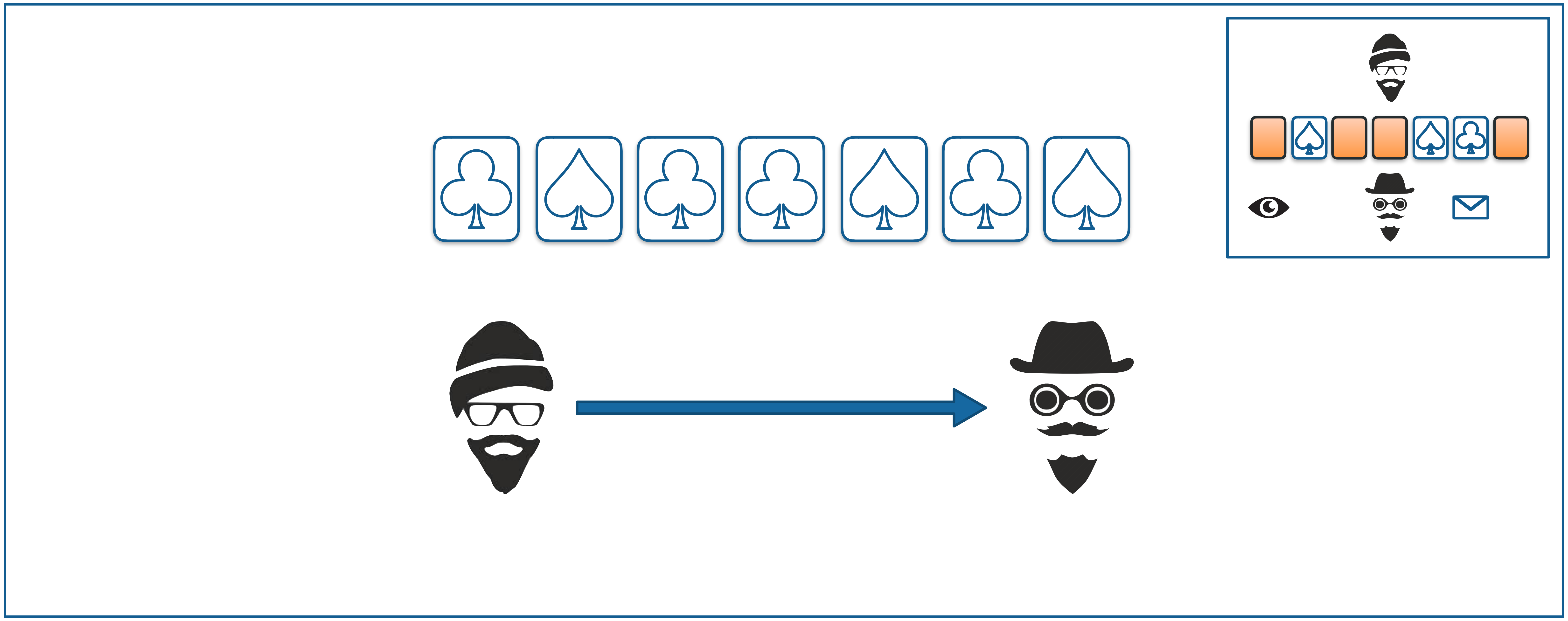
Prove(, i) = $\pi\{$The i'th bit of VPRG( ) using the seed in is $\}$

Verify(, i, π,) = yes / no

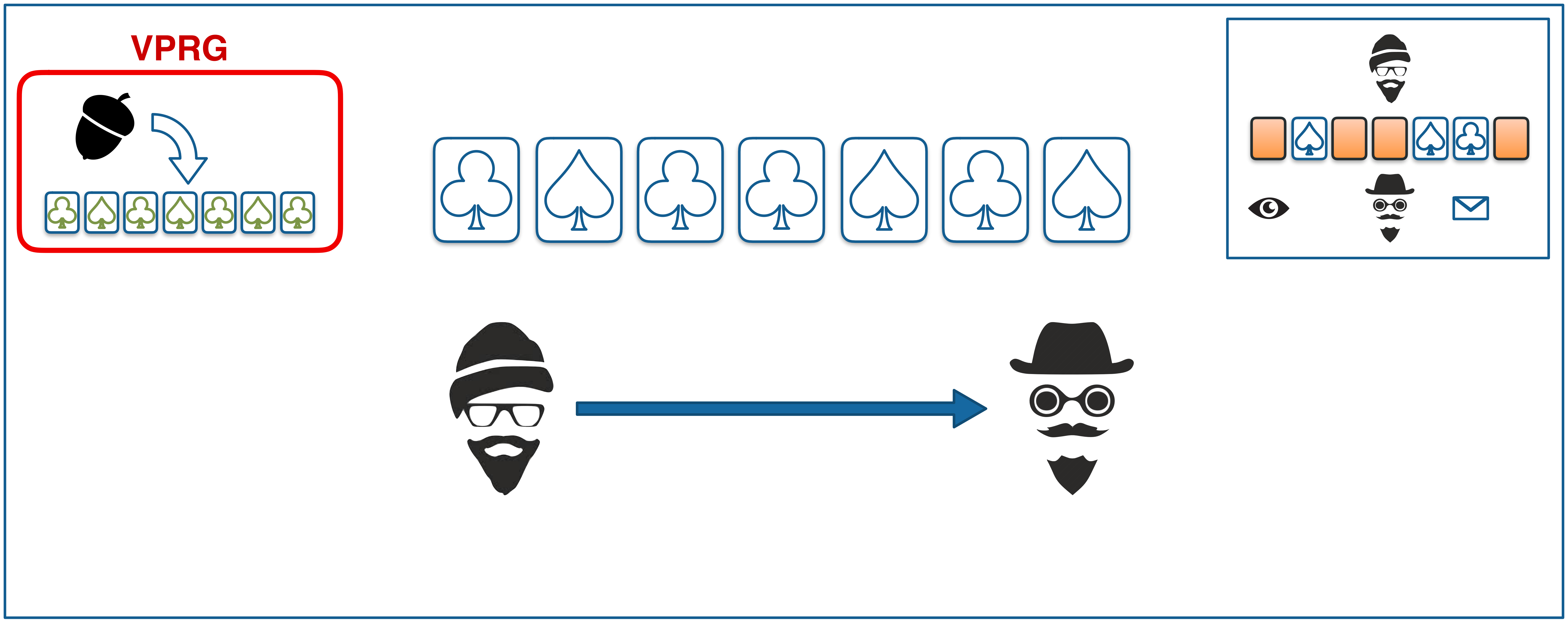
- Is short
- The proof leaks nothing more about
- **The proof is sound in a strong sense**

1. Every *is in the image of* VPRG(.)
2. For *every possible* , there is a *unique* associated
3. Proofs of opening to bits inconsistent with *do not exist*

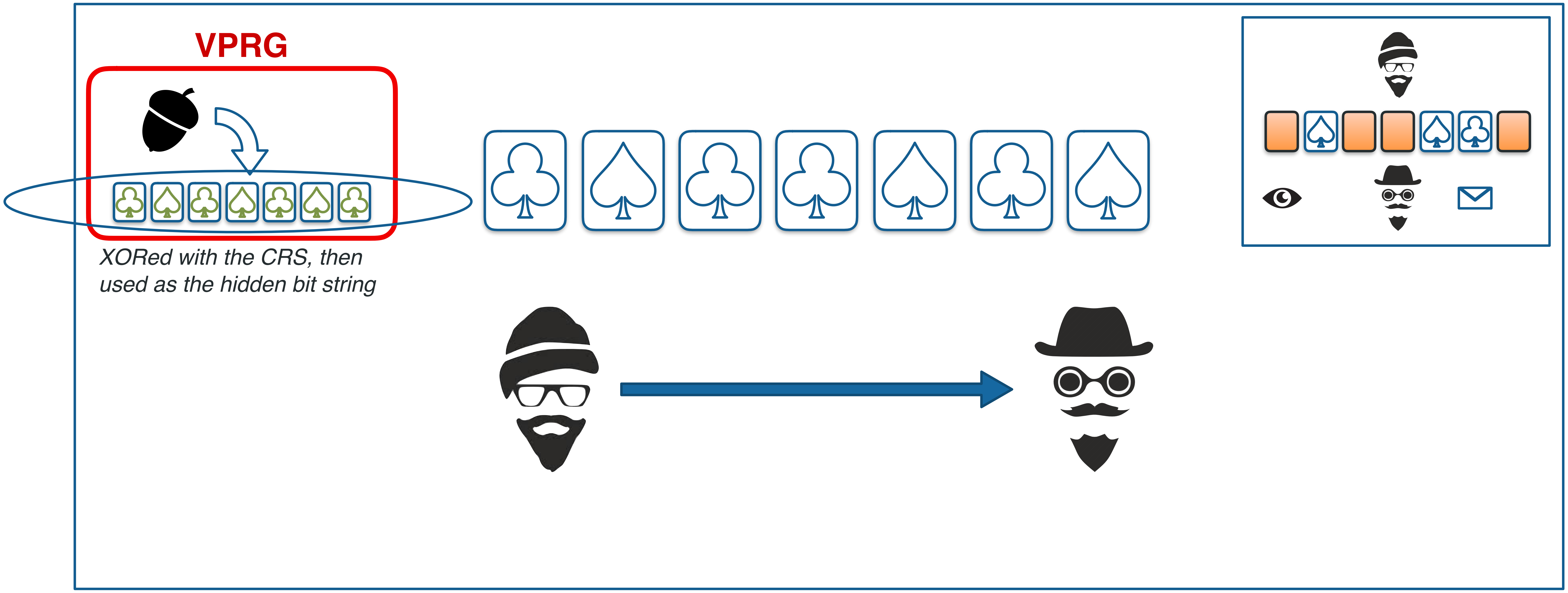
Building NIZKs from VPRGs



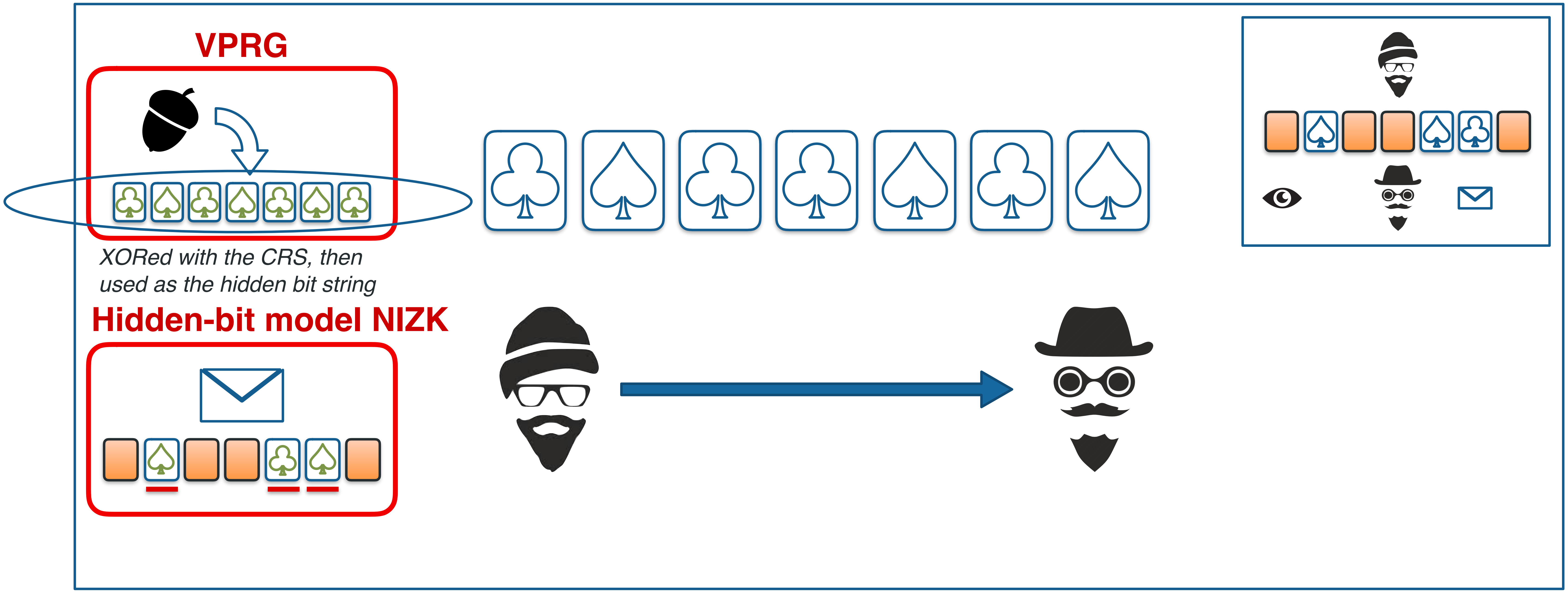
Building NIZKs from VPRGs



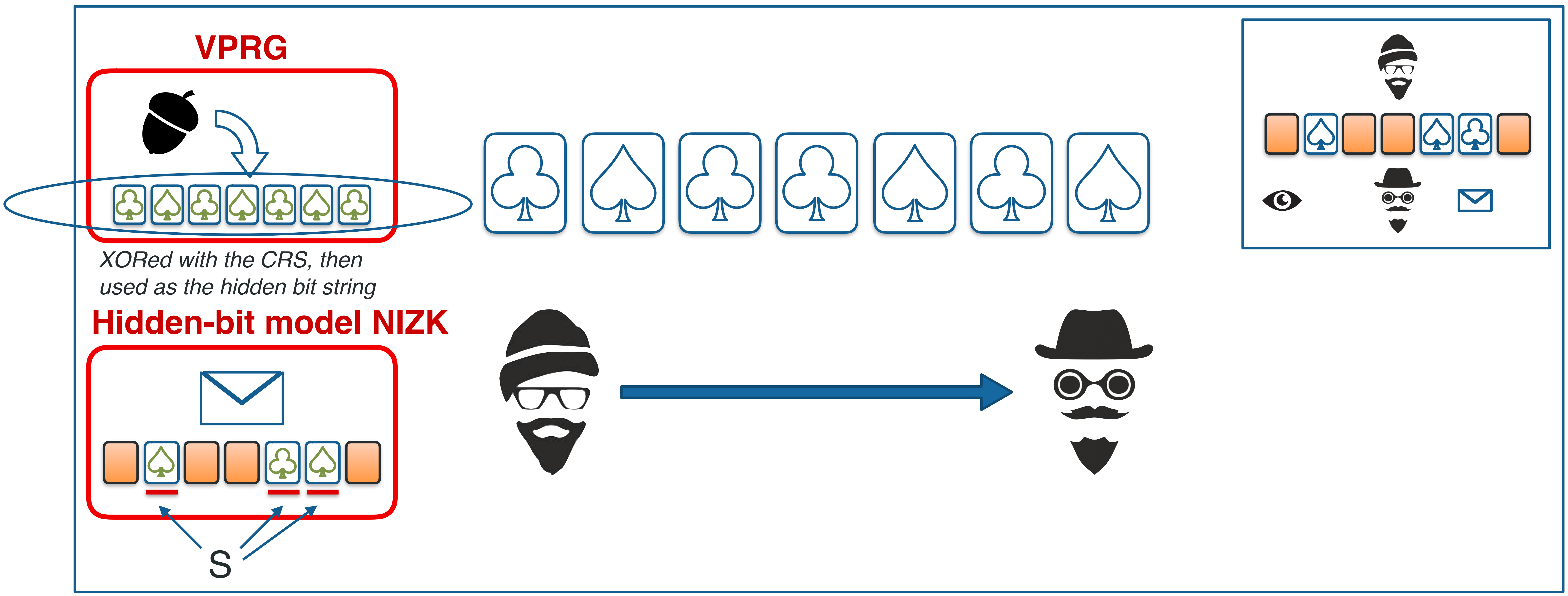
Building NIZKs from VPRGs



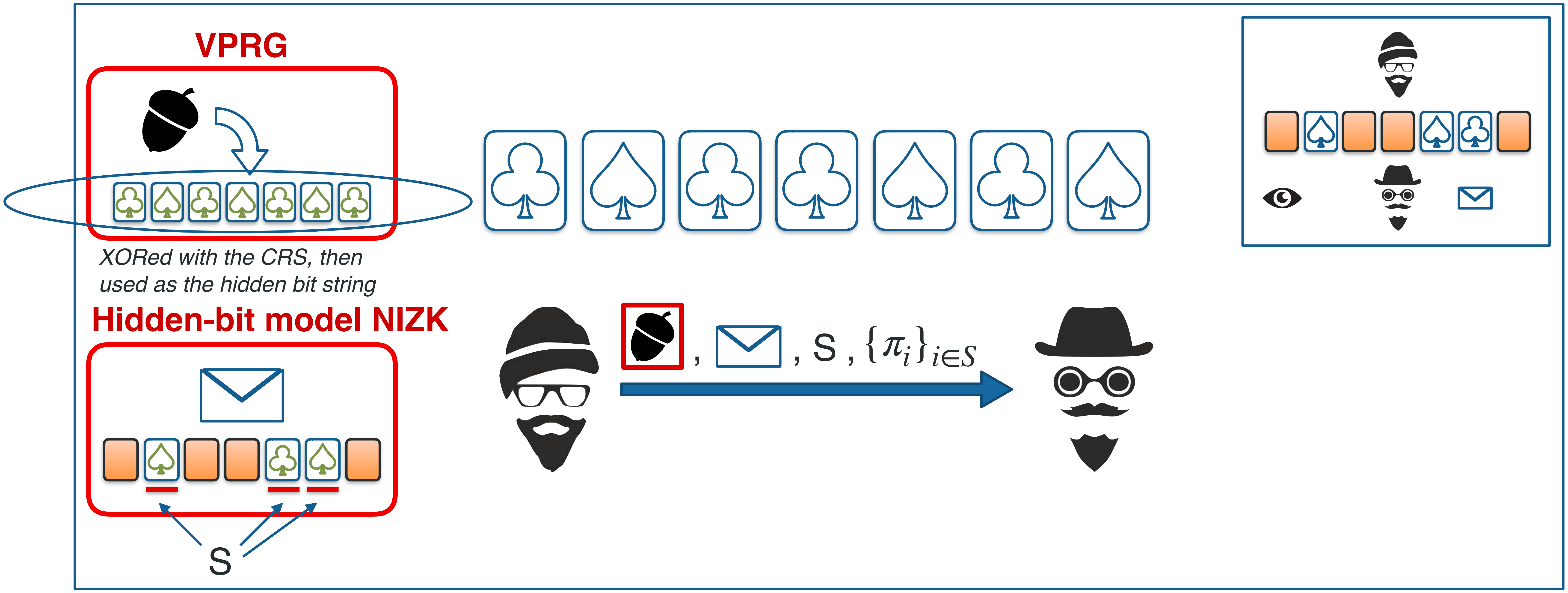
Building NIZKs from VPRGs



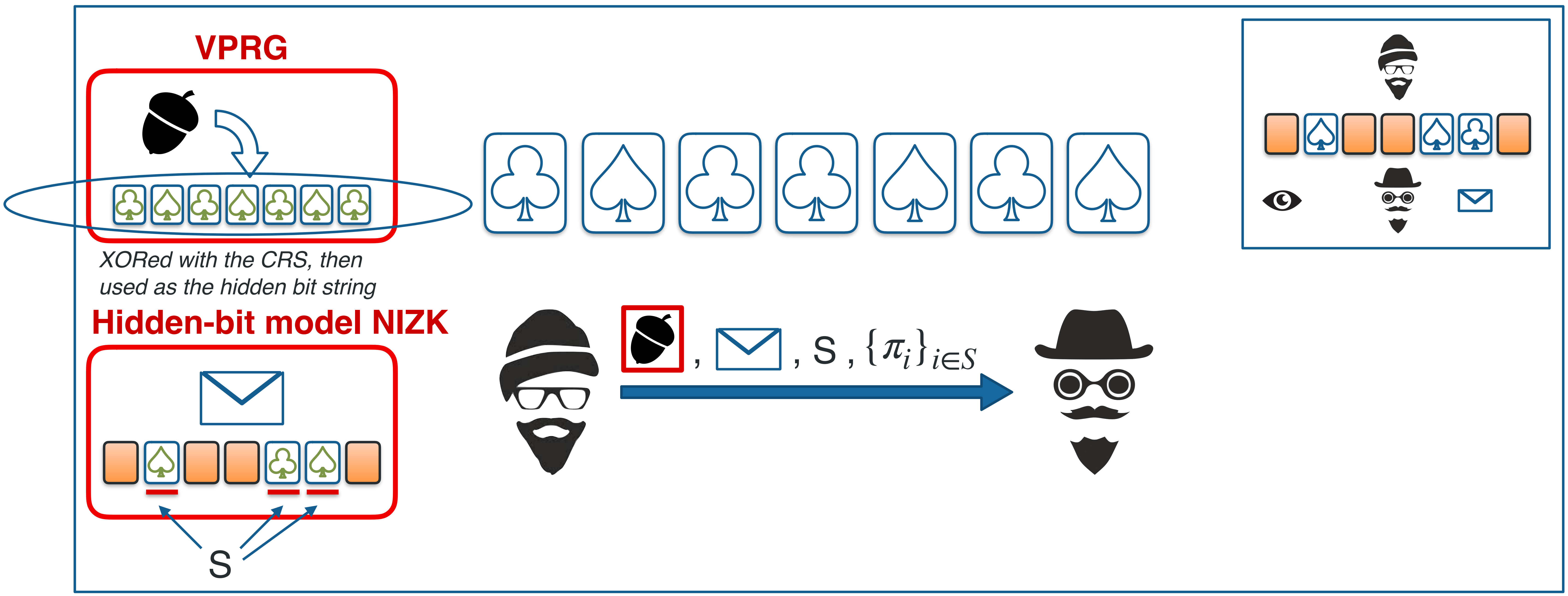
Building NIZKs from VPRGs



Building NIZKs from VPRGs

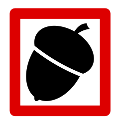
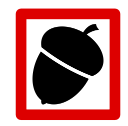


Building NIZKs from VPRGs

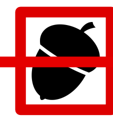
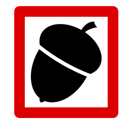


1. Every is in the image of VPRG(.)
2. For *every possible* , there is a *unique* associated
3. Proofs of opening to bits inconsistent with *do not exist*

Relaxing VPRGs

1. Every  *is in the image of* VPRG(.)
2. For *every possible*  , there is a *unique* associated ♧♤♧♧♤♧♤
3. Proofs of opening to bits inconsistent with ♧♤♧♧♤♧♤ *do not exist*

Relaxing VPRGs

1. ~~Every  *is in the image of* VPRG(.)~~
2. For *every possible*  , there is a *unique* associated ♧♤♧♧♤♧♤
3. Proofs of opening to bits inconsistent with ♧♤♧♧♤♧♤ *do not exist*

Relaxing VPRGs

~~1. Every is in the image of VPRG(.)~~

2. For *every possible* , there is a *unique* associated ♧♤♧♧♤♧♤

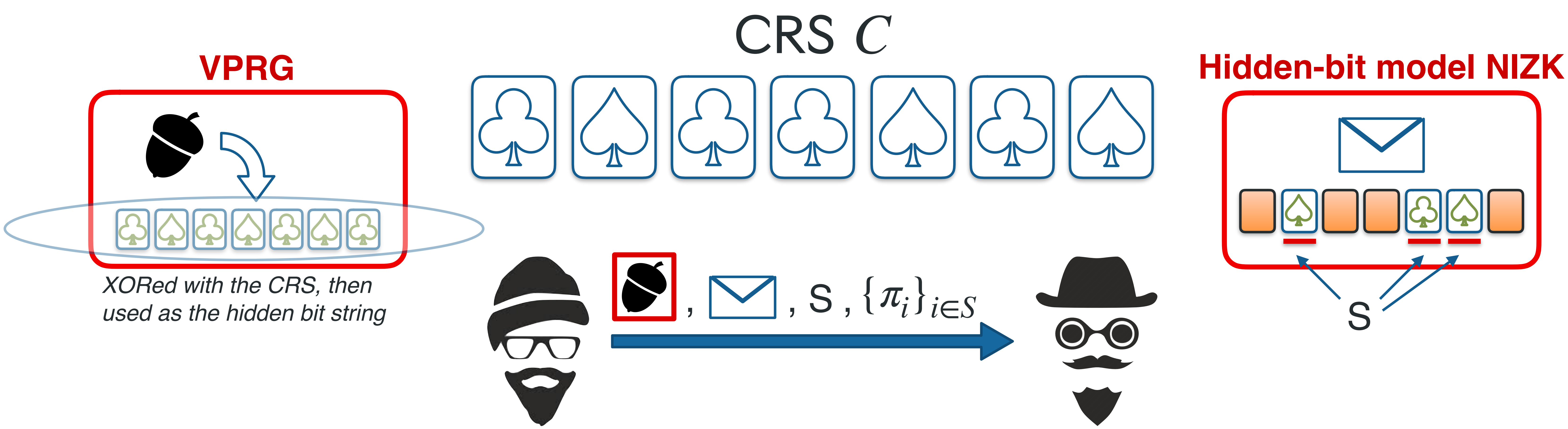
3'. Proofs of opening to bits inconsistent with ♧♤♧♧♤♧♤ *are hard to find*

Relaxing VPRGs

~~1. Every *is in the image of* VPRG(.)~~
2. For *every possible*, there is a *unique* associated ♧♤♧♧♤♧♤
3'. Proofs of opening to bits inconsistent with ♧♤♧♧♤♧♤ *are hard to find*
4. *is short*

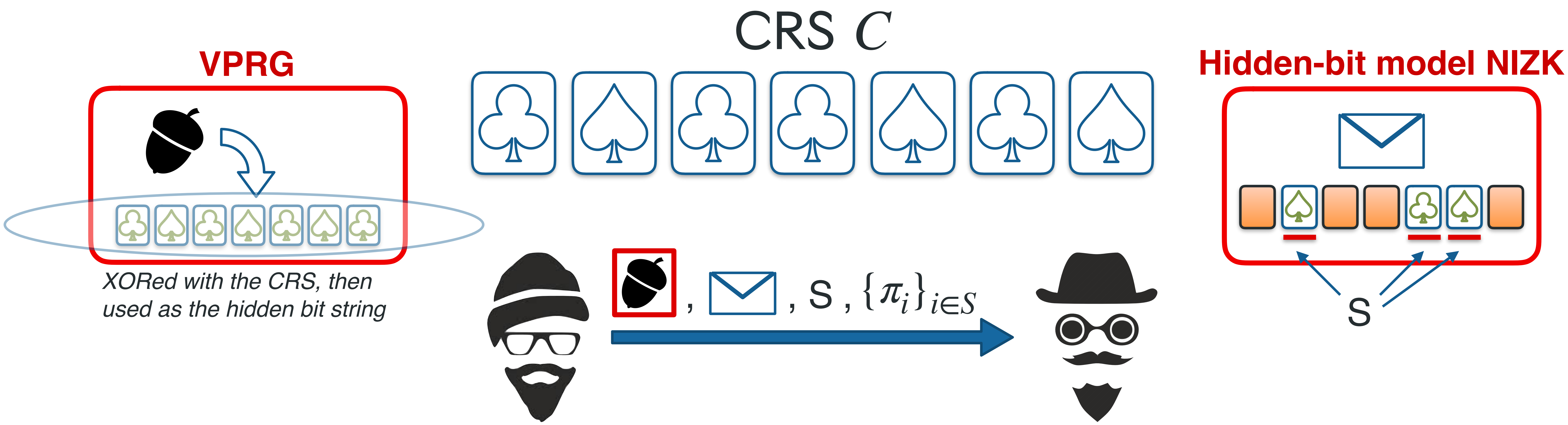
Relaxing VPRGs

1. ~~Every is in the image of VPRG(.)~~
2. For *every possible* , there is a *unique* associated
3'. Proofs of opening to bits inconsistent with *are hard to find*
4. *is short*



Relaxing VPRGs

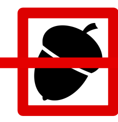
~~1. Every *is in the image of* VPRG(.)~~
2. For *every possible* , there is a *unique* associated
3'. Proofs of opening to bits inconsistent with *are hard to find*
4. *is short*

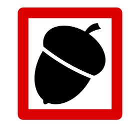


Proof Idea:

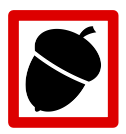
- C is 'close to a bad string' if $\exists$, Ext( ) $\oplus$ C is bad
- Proof accepted iff inconsistent opening OR the CRS is « close to a bad string » (requires (2))
- Inconsistent opening → contradiction to VPRG (3')
- Since is short, few CRS are close to a bad string.

Relaxing VPRGs

~~1. Every~~  ~~*is in the image of* VPRG(.)~~

2. For *every possible*  , there is a *unique* associated ♧♤♧♧♤♧♤

3’. Proofs of opening to bits inconsistent with ♧♤♧♧♤♧♤ *are hard to find*

4.  *is short*

How does that help?

Relaxing VPRGs

1. ~~Every *is in the image of* VPRG(.)~~
2. For *every possible* , there is a *unique* associated
3'. Proofs of opening to bits inconsistent with *are hard to find*
4. *is short*

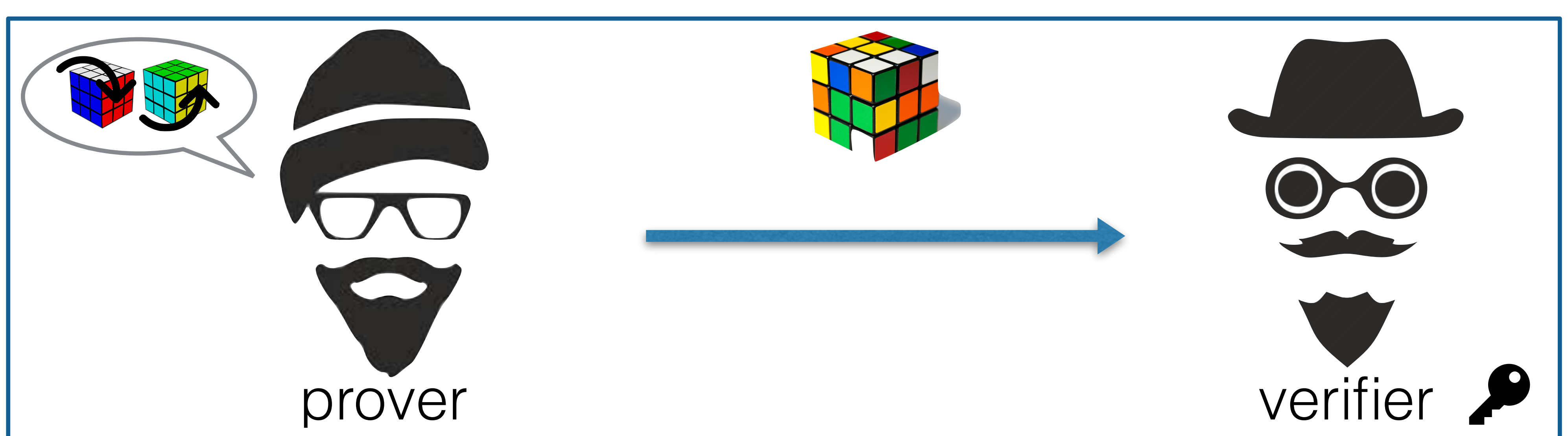
How does that help?

~~(1)~~ allows for lattice-based VPRGs

For typical LWE-based commitments, there are many invalid commitments indistinguishable from valid ones

(3') allows for designated-verifier variants

Since accepting incorrect proofs always exist in the DV setting



Instantiation 1: DVPRG from CDH

CDH over a group $\mathbb{G}$ states that given random g, g^a, g^b, it is hard to find g^{ab}

Instantiation 1: DVPRG from CDH

CDH over a group $\mathbb{G}$ states that given random g, g^a, g^b, it is hard to find g^{ab}

[CKS08], gap twin-CDH: given random g, g^a, g^b, g^c, it is hard to find g^{ab}, g^{ac} *even given an oracle for the twin-DDH problem*

CDH $\Leftrightarrow$ gap twin-CDH *using some secret 'twin-DDH checking key'*

Instantiation 1: DVPRG from CDH

CDH over a group $\mathbb{G}$ states that given random g, g^a, g^b, it is hard to find g^{ab}

[CKS08], gap twin-CDH: given random g, g^a, g^b, g^c, it is hard to find g^{ab}, g^{ac}
even given an oracle for the twin-DDH problem

CDH $\Leftrightarrow$ gap twin-CDH *using some secret 'twin-DDH checking key'*

[GL89]: explicit predicate B(.) such that given random g, g^a, g^b, g^c, it is hard to find $B(g^{ab}, g^{ac})$ with probability >> 1/2 even given an oracle for the twin-DDH problem

Equivalent to CDH

Instantiation 1: DVPRG from CDH

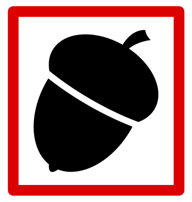
CDH over a group $\mathbb{G}$ states that given random g, g^a, g^b, it is hard to find g^{ab}

[CKS08], gap twin-CDH: given random g, g^a, g^b, g^c, it is hard to find g^{ab}, g^{ac}
even given an oracle for the twin-DDH problem

CDH $\Leftrightarrow$ gap twin-CDH *using some secret 'twin-DDH checking key'*

[GL89]: explicit predicate B(.) such that given random g, g^a, g^b, g^c, it is hard to find $B(g^{ab}, g^{ac})$ with probability >> 1/2 even given an oracle for the twin-DDH problem

Equivalent to CDH

= 

Instantiation 1: DVPRG from CDH

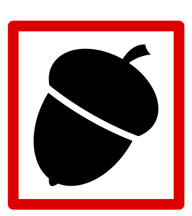
CDH over a group $\mathbb{G}$ states that given random g, g^a, g^b, it is hard to find g^{ab}

[CKS08], gap twin-CDH: given random g, g^a, g^b, g^c, it is hard to find g^{ab}, g^{ac}
even given an oracle for the twin-DDH problem

CDH $\Leftrightarrow$ gap twin-CDH *using some secret 'twin-DDH checking key'*

[GL89]: explicit predicate B(.) such that given random g, g^a, g^b, g^c, it is hard to find $B(g^{ab}, g^{ac})$ with probability >> 1/2 even given an oracle for the twin-DDH problem

Equivalent to CDH

= 

= public parameters

Instantiation 1: DVPRG from CDH

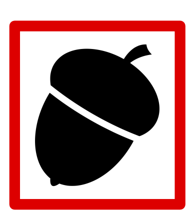
CDH over a group $\mathbb{G}$ states that given random g, g^a, g^b, it is hard to find g^{ab}

[CKS08], gap twin-CDH: given random g, g^a, g^b, g^c, it is hard to find g^{ab}, g^{ac} *even given an oracle for the twin-DDH problem*

CDH $\Leftrightarrow$ gap twin-CDH *using some secret 'twin-DDH checking key'*

[GL89]: explicit predicate B(.) such that given random g, g^a, g^b, g^c, it is hard to find $B(g^{ab}, g^{ac})$ with probability >> 1/2 even given an oracle for the twin-DDH problem

Equivalent to CDH

= 

= public parameters

= pseudorandom bit associated to with respect to

Instantiation 1: DVPRG from CDH

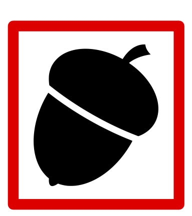
CDH over a group $\mathbb{G}$ states that given random g, g^a, g^b, it is hard to find g^{ab}

[CKS08], gap twin-CDH: given random g, g^a, g^b, g^c, it is hard to find g^{ab}, g^{ac}
even given an oracle for the twin-DDH problem

CDH $\Leftrightarrow$ gap twin-CDH *using some secret 'twin-DDH checking key'*

[GL89]: explicit predicate B(.) such that given random g, g^a, g^b, g^c, it is hard to find $B(g^{ab}, g^{ac})$
with probability >> 1/2 even given an oracle for the twin-DDH problem

Equivalent to CDH

= 

= public parameters

= pseudorandom bit associated to with respect to

Proof: g^{ab}, g^{ac}
+ twin-DDH check

Instantiation 1: DVPRG from CDH

CDH over a group $\mathbb{G}$ states that given random g, g^a, g^b, it is hard to find g^{ab}

[CKS08], gap twin-CDH: given random g, g^a, g^b, g^c, it is hard to find g^{ab}, g^{ac}
even given an oracle for the twin-DDH problem

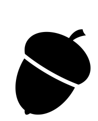
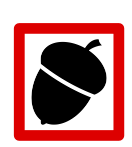
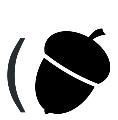
CDH $\Leftrightarrow$ gap twin-CDH *using some secret 'twin-DDH checking key'*

[GL89]: explicit predicate B(.) such that given random g, g^a, g^b, g^c, it is hard to find $B(g^{ab}, g^{ac})$ with probability >> 1/2 even given an oracle for the twin-DDH problem

Equivalent to CDH

Public parameters: $\mathbb{G}, g, (g^{a_1}, g^{b_1}, \cdots, g^{a_n}, g^{b_n}) = (u_1, v_1, \cdots, u_n, v_n)$

Secret verification key: $(\lambda_1, \cdots, \lambda_n)$ and $(K_1, \cdots, K_n) = (a_1 + \lambda_1 b_1, \cdots, a_n + \lambda_n b_n)$

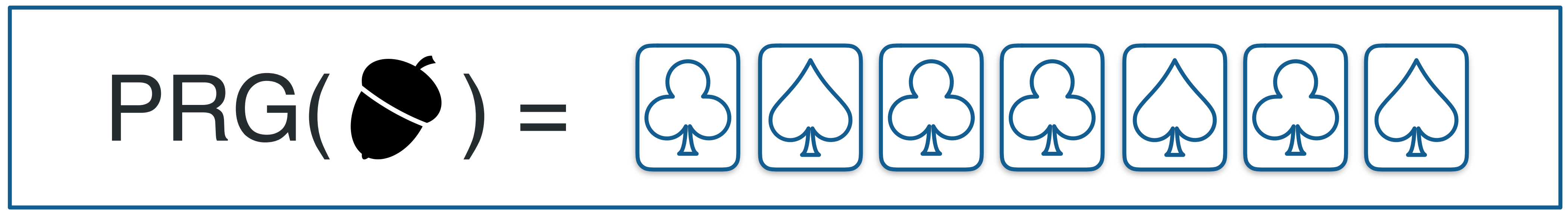
DVPRG:  $= r$,  $= g^r$, DVPRG() = $B({u_1}^r, {v_1}^r), \cdots, B({u_n}^r, {v_n}^r)$

Proof: $\pi = (u_i^r, v_i^r) = (\pi_0, \pi_1)$

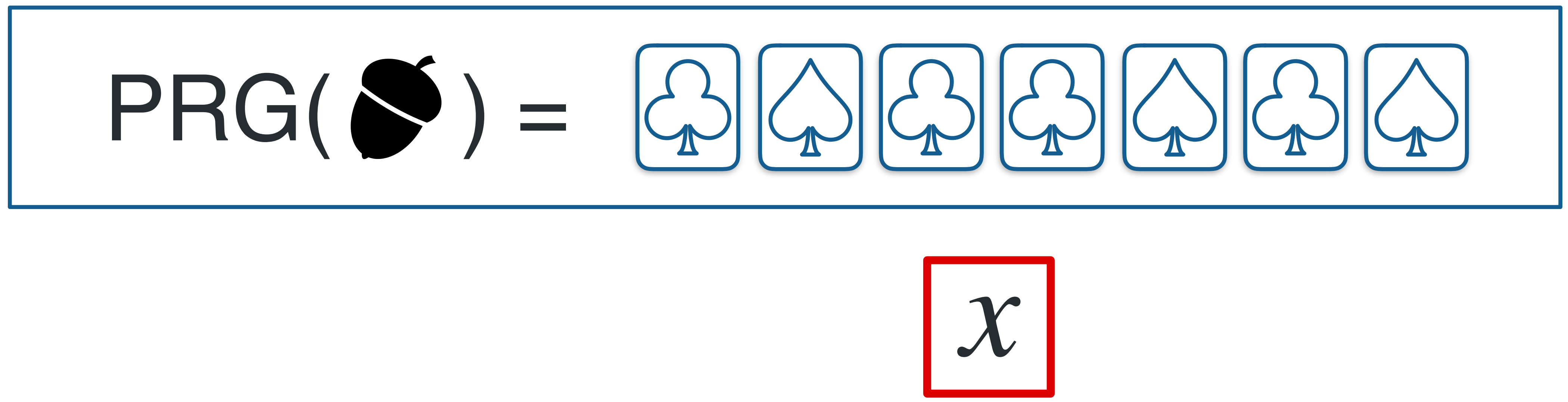
Verification: check that $B(\pi_0, \pi_1) = b$ and $\pi_0^{\lambda_i} \pi_1 = (g^r)^{K_i}$

Instantiation 2: VPRG from LWE+NIWI

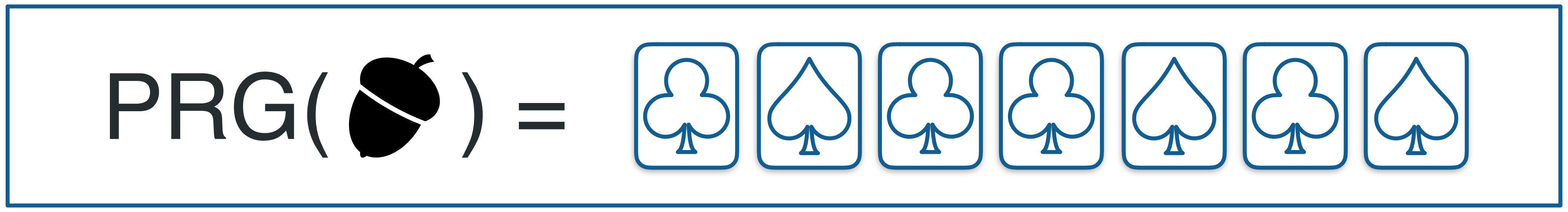
Instantiation 2: VPRG from LWE+NIWI



Instantiation 2: VPRG from LWE+NIWI

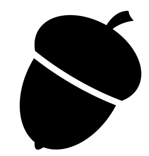
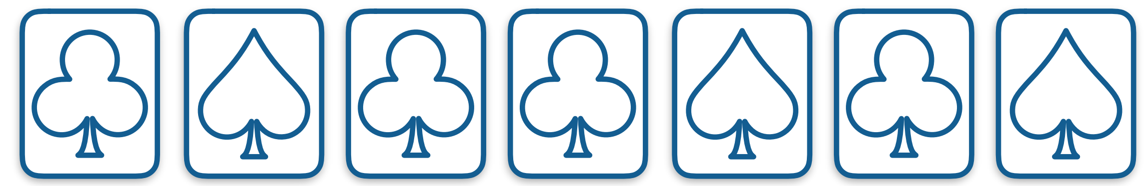


Instantiation 2: VPRG from LWE+NIWI



Hiding $\boxed{x}$

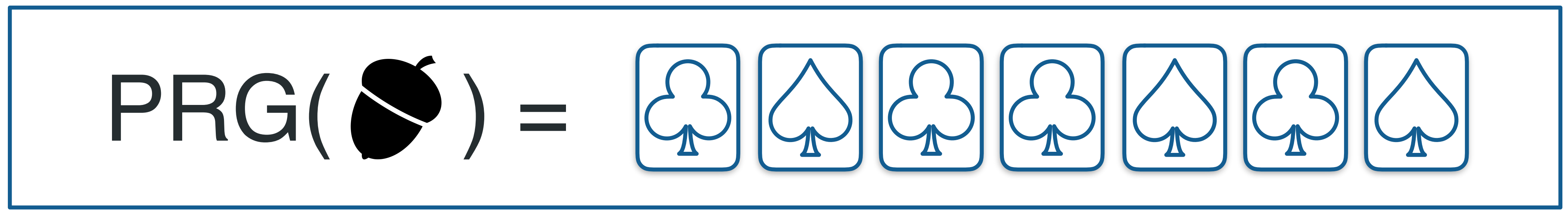
Instantiation 2: VPRG from LWE+NIWI

PRG() = 

Hiding $\boxed{x}$

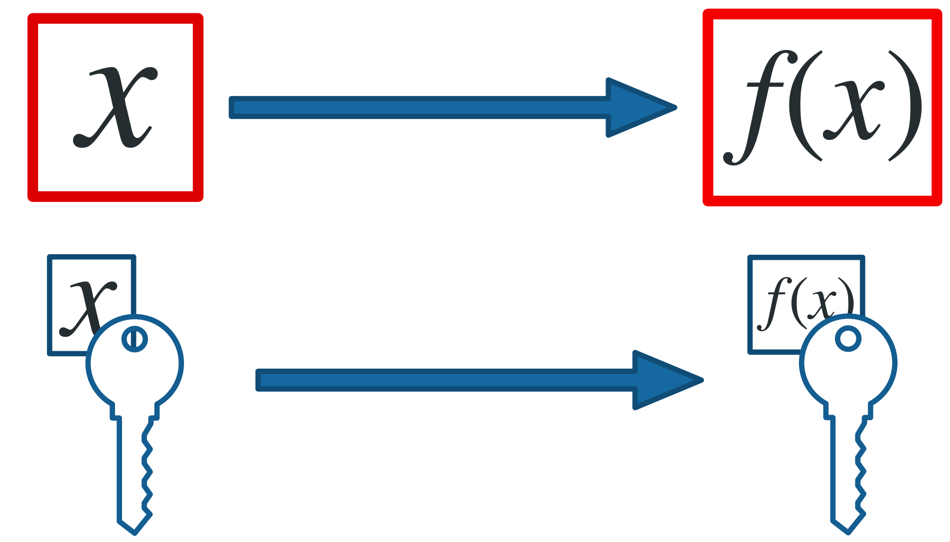
Binding 

Instantiation 2: VPRG from LWE+NIWI



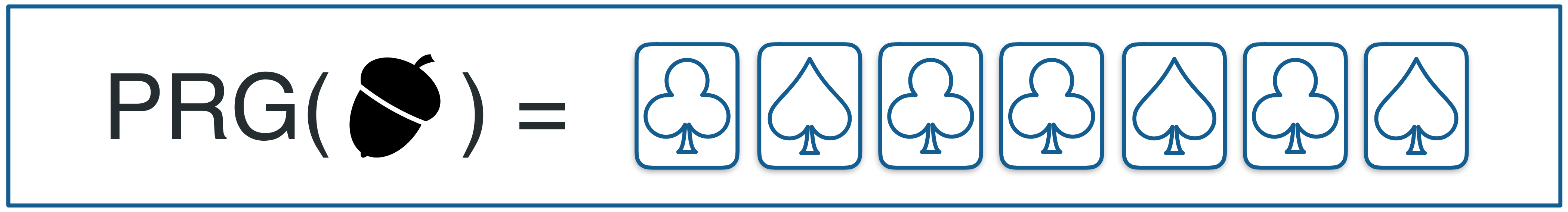
Hiding

Binding



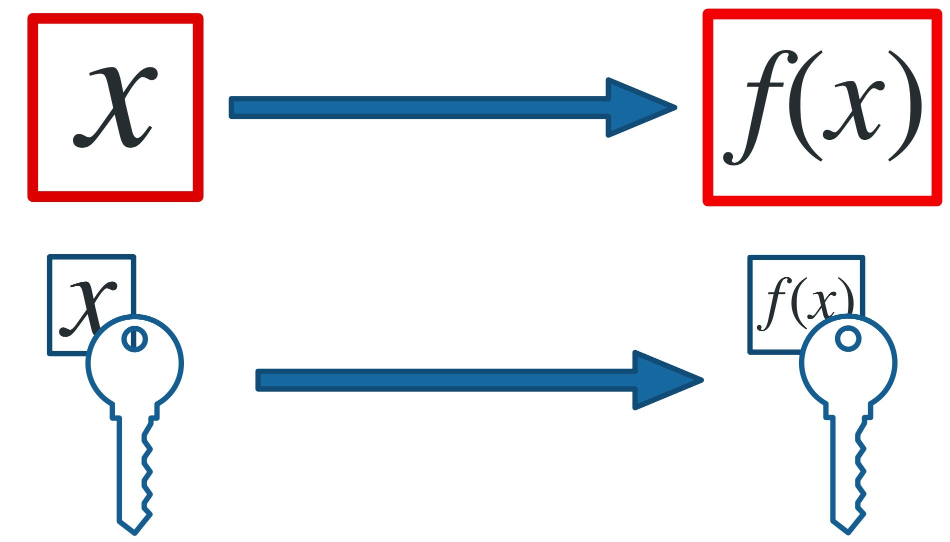
Fully homomorphic

Instantiation 2: VPRG from LWE+NIWI

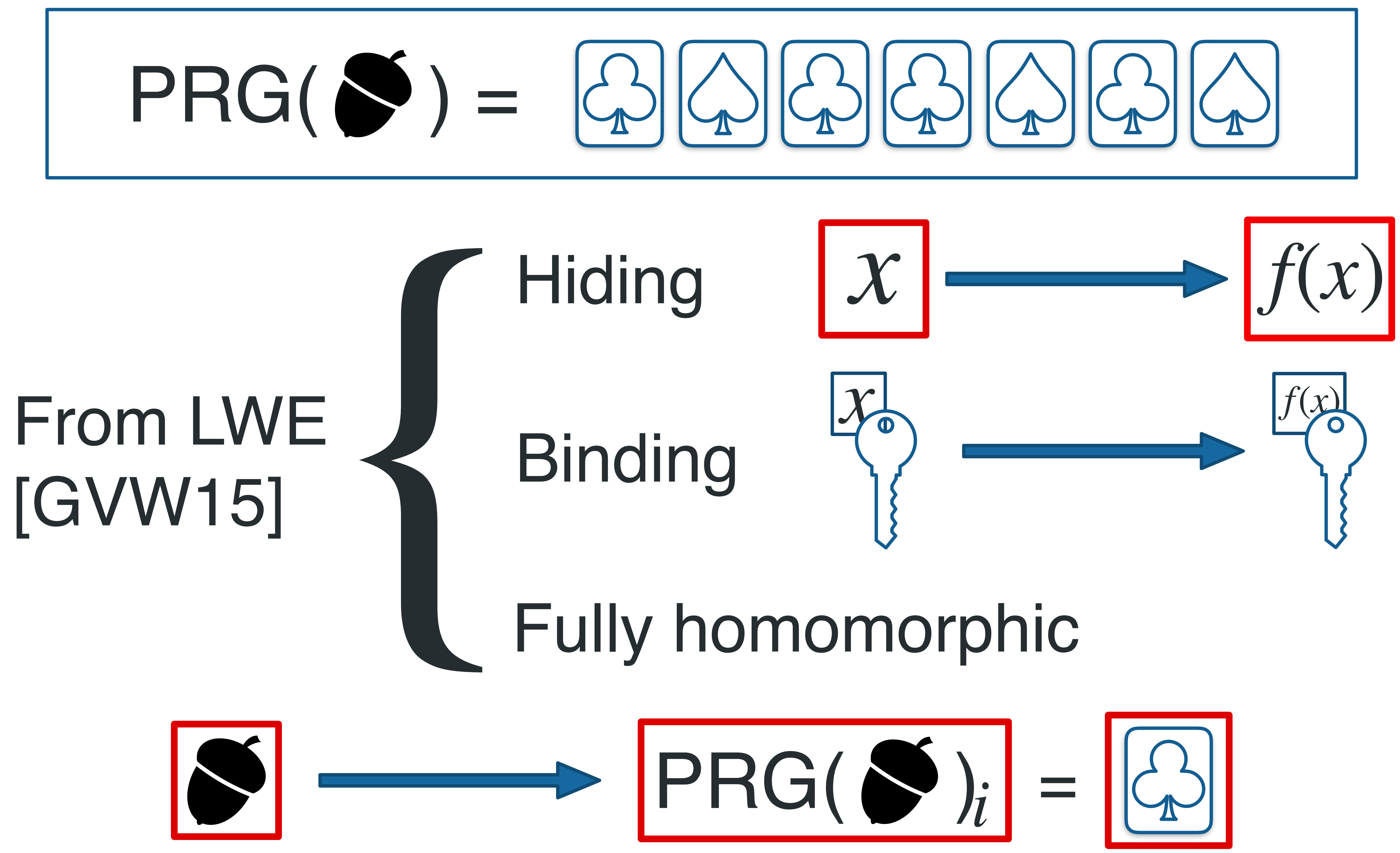


From LWE [GVW15]

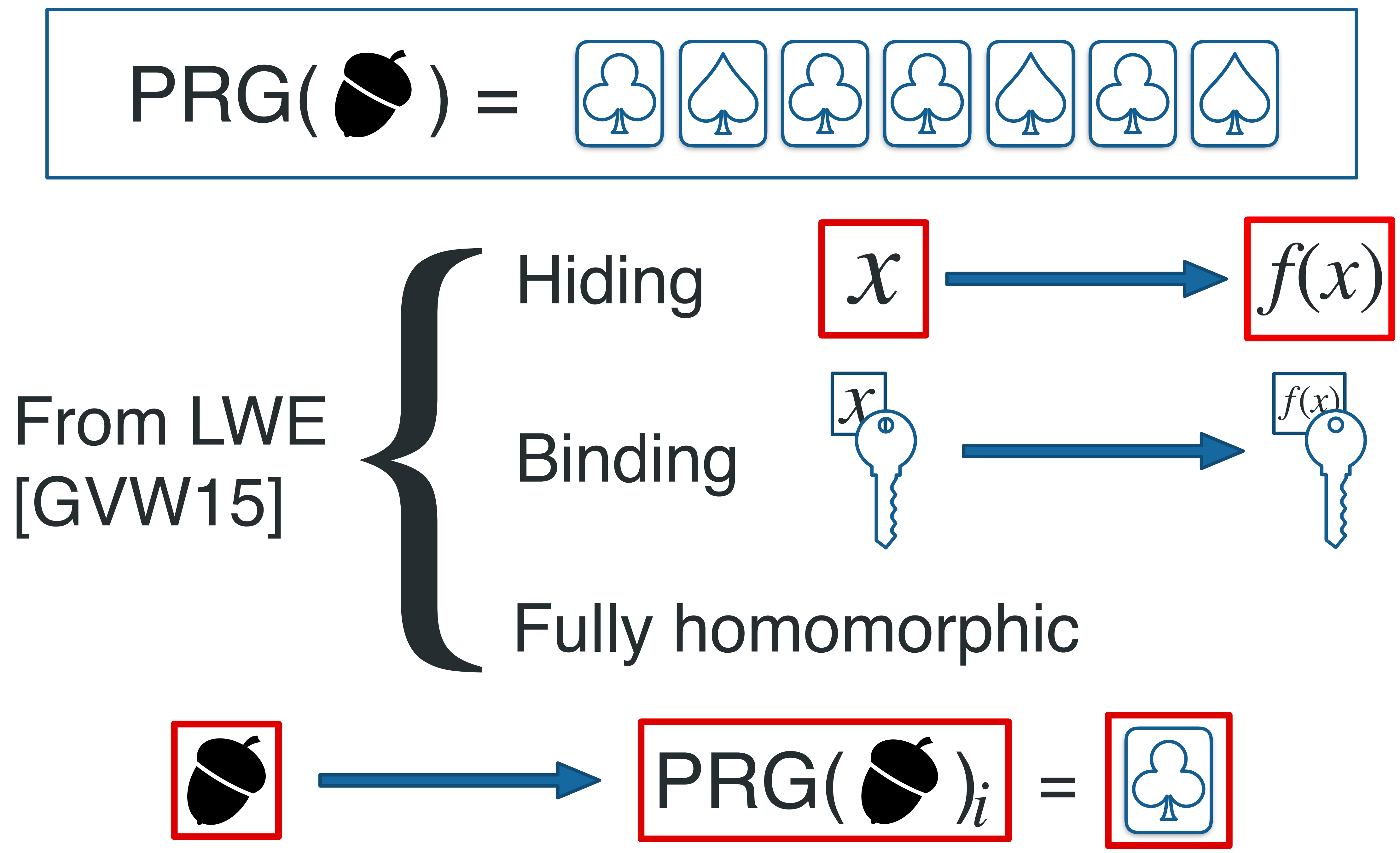
- Hiding
- Binding
- Fully homomorphic



Instantiation 2: VPRG from LWE+NIWI

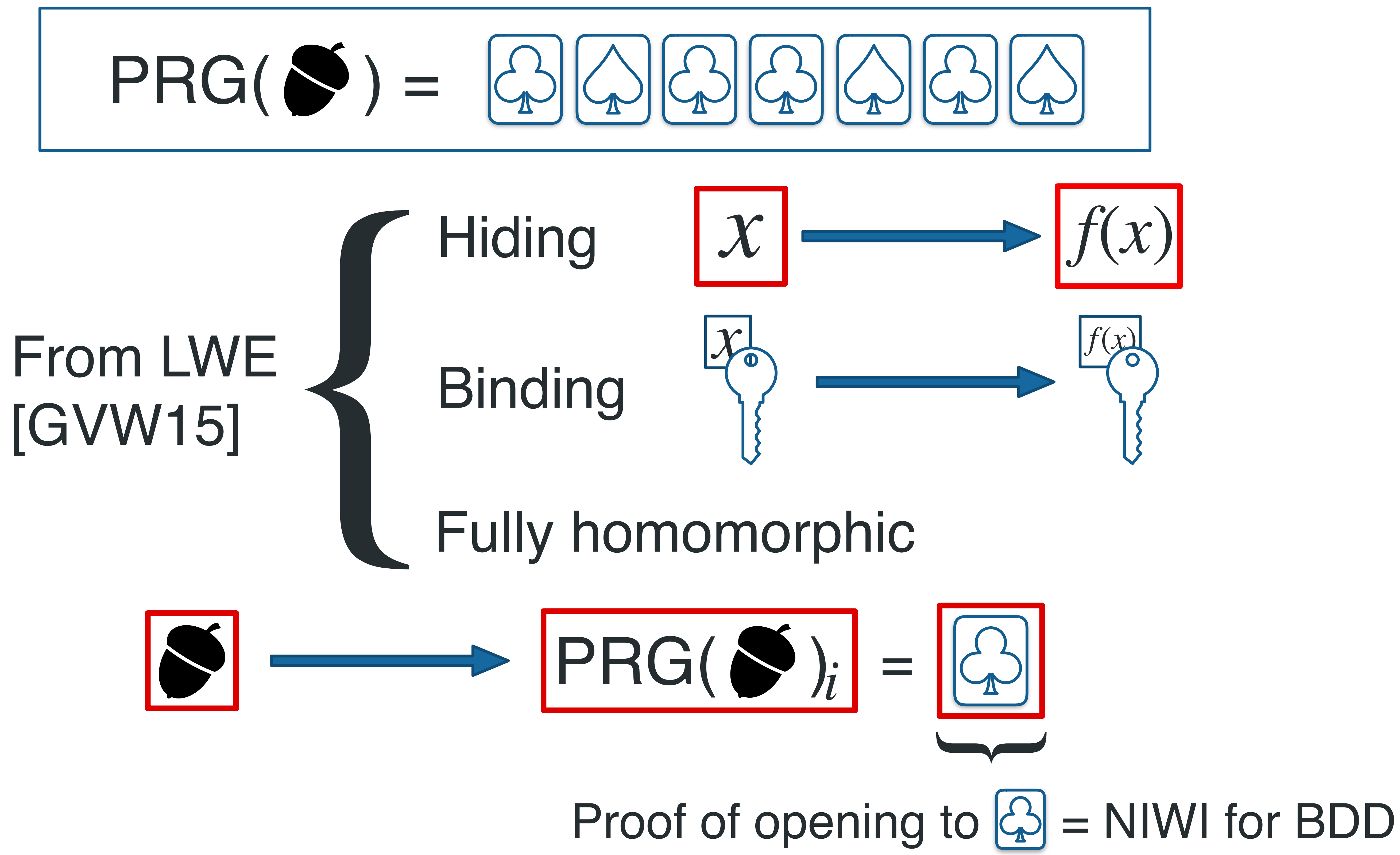


Instantiation 2: VPRG from LWE+NIWI



1. ~~Every is in the image of VPRG(.)~~
2. For *every possible* , there is a *unique* associated
3'. Proofs of opening to bits inconsistent with *are hard to find*
4. *is short*

Instantiation 2: VPRG from LWE+NIWI

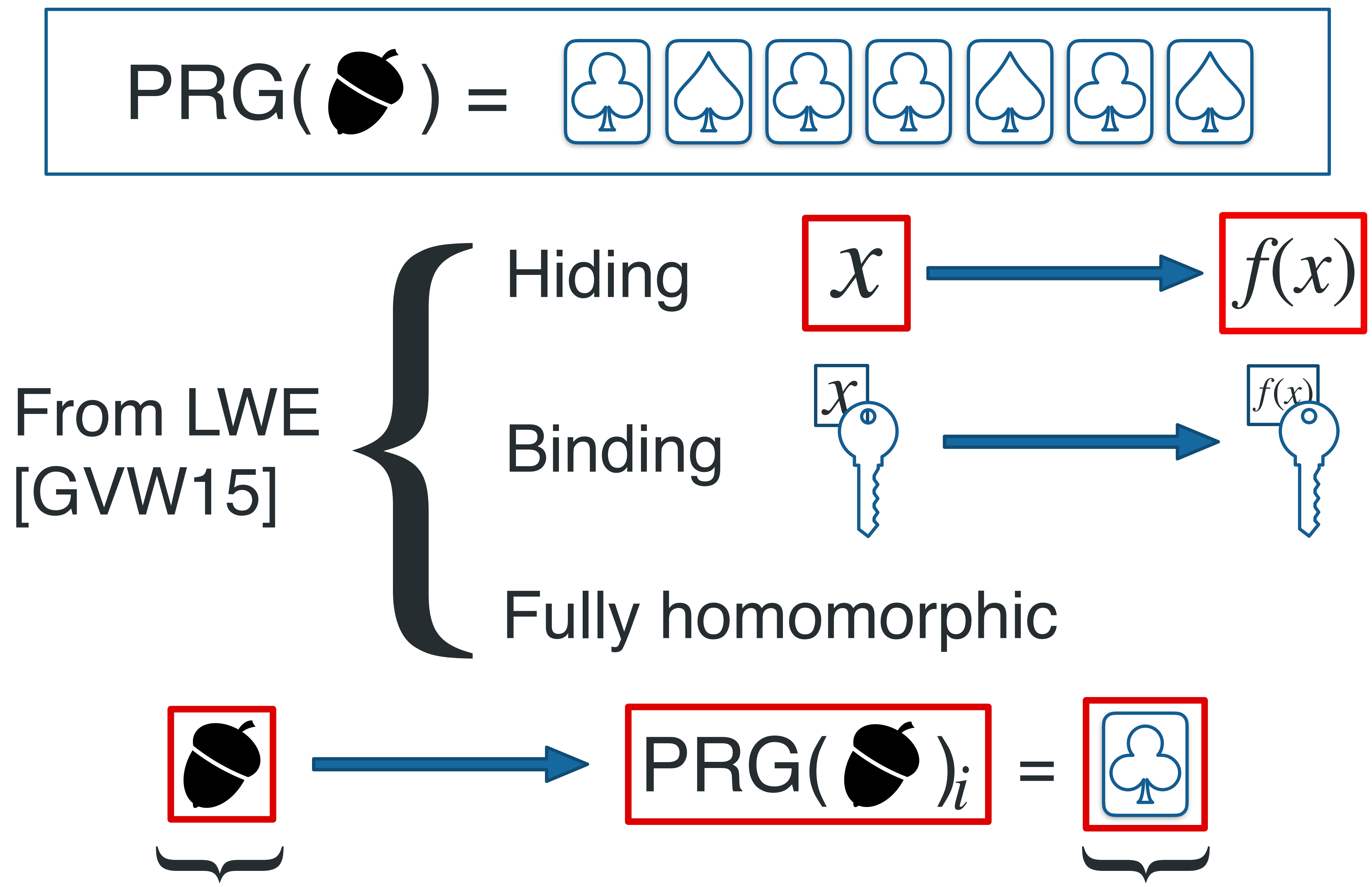


1. ~~Every is in the image of VPRG(.)~~
2. For *every possible* , there is a *unique* associated

3’. Proofs of opening to bits inconsistent with *are hard to find*

4. *is short*

Instantiation 2: VPRG from LWE+NIWI



Proof of validity = NIZK for BDD. Proof of opening to = NIWI for BDD.

1. ~~Every is in the image of VPRG(.)~~
2. For *every possible* , there is a *unique* associated
3'. Proofs of opening to bits inconsistent with *are hard to find*
4. *is short*

Summary

We obtain two new constructions:

1) A DVNIZK for NP under the CDH assumption

First direct indication that DVNIZK with unbounded soundness are actually easier to build than standard NIZK

2) A (DV)NIZK for NP assuming LWE and the existence of a (DV)NIWI for BDD

Improving over, and considerably simplifying, the recent result of [RR18] which required a NIZK for BDD.

by relaxing [DN00]'s VPRGs, generalizing to DVPRGs, showing that it still suffices to construct (DV)NIZKs by instantiating the hidden-bit model, and providing new (D)PRGs instantiations.

Thanks for your attention

Questions?