Strong Key Derivation from Biometrics

Benjamin Fuller,
Boston University/MIT Lincoln Laboratory

Privacy Enhancing Technologies for Biometrics, Haifa

January 15, 2015

Based on three works:

- *Computational Fuzzy Extractors* [FullerMengReyzin13]
- *When are Fuzzy Extractors Possible?* [FullerSmithReyzin14]
- *Key Derivation from Noisy Sources with More Errors than Entropy* [CanettiFullerPanethSmithReyzin14]

Key Derivation from Noisy Sources

High-entropy sources are often noisy

- Initial reading $w_0 \neq$ later reading reading w_1
- Consider sources $w_0 = a_1, \ldots, a_k$, each symbol a_i over alphabet Z
- Assume a bound distance: $d(w_0, w_1) \leq t$

Biometric Data

$d(w_0, w_1)$=# of symbols in that differ

$d(w_0, w_1)=4$

w_0	A	B	C	A	D	B	E	F	A	A
w_1	A	G	C	A	B	B	E	F	C	B

Key Derivation from Noisy Sources

High-entropy sources are often noisy

- Initial reading $w_0 \neq$ later reading reading w_1
- Consider sources $w_0 = a_1, \ldots, a_k$, each symbol a_i over alphabet Z
- Assume a bound distance: $d(w_0, w_1) \leq t$

Goal: derive a stable cryptographically strong output

- Want w_0, w_1 to map to same output
- The output should look uniform to the adversary

Biometric Data

Goal of this talk: produce good outputs for sources we couldn't handle before

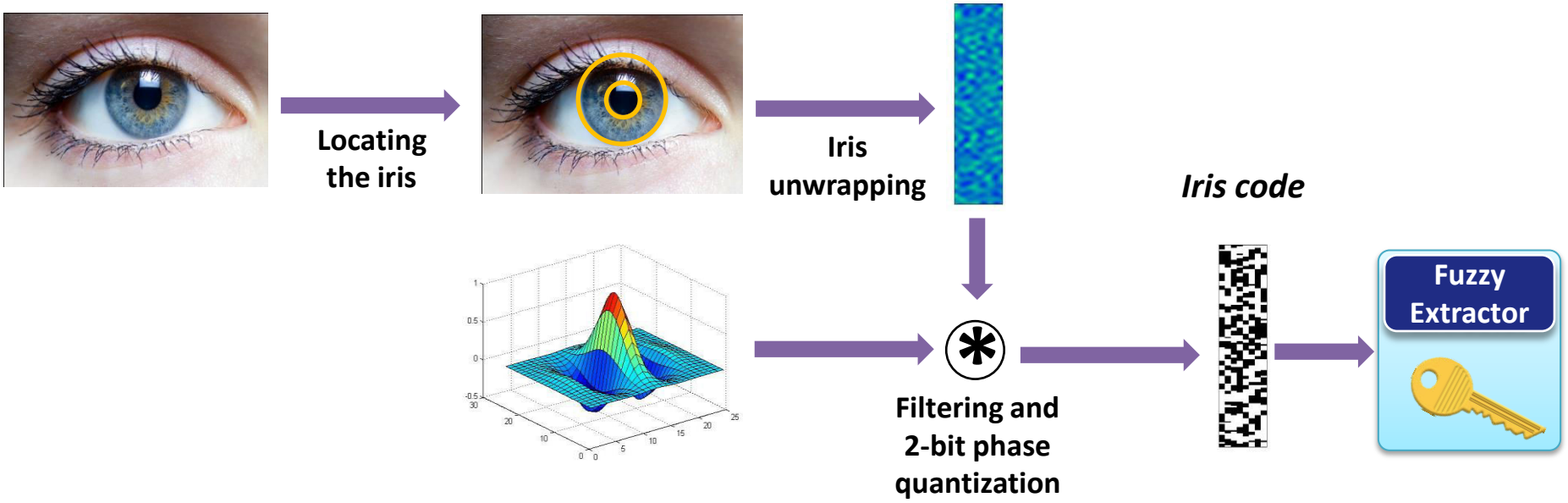
Biometrics

- Measure unique physical phenomenon
- Unique, collectable, permanent, universal
- Repeated readings exhibit significant noise
- Uniqueness/Noise vary widely
- Human iris believed to be “best”

[Daugman04], [PrabhakarPankantiJain03]

Theoretic work, with iris in mind

Iris Codes [Daugman04]

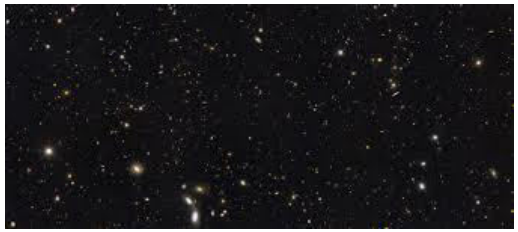


- Iris code: sequence of quantized wavelets (computed at different positions)
- Daugman's transform is 2048 bits long
- Entropy estimate 249 bits
- Error rate depends on conditions, user applications 10%

Two Physical Processes

Uncertainty

w_0 – create a new biometric, take initial reading



w_0

w_1

Errors

w_1 – take new reading from a fixed person

Two readings may not be subject to same noise. Often less error in original reading

Key Derivation from Noisy Sources

Interactive Protocols

[Wyner75] … [BennettBrassardRobert85,88] …lots of work…

w w_0 w_1

Parties agree on cryptographic key

User must store initial reading w_0 at server

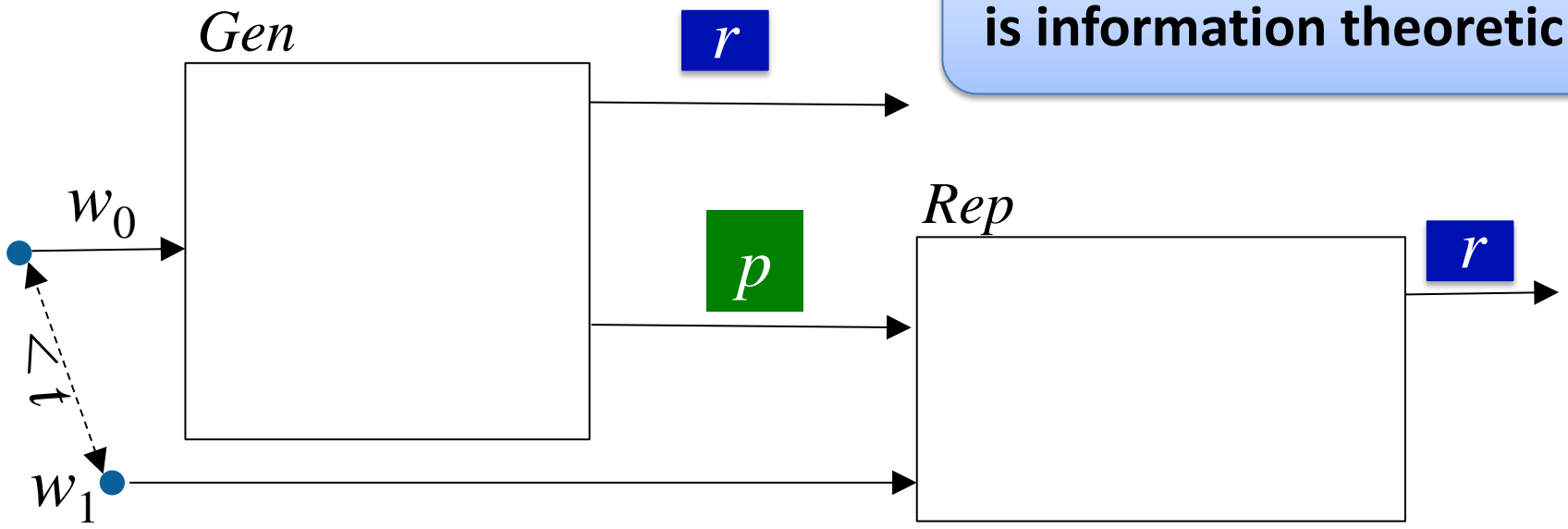
Not appropriate for user authenticating to device

Fuzzy Extractors: Functionality

[JuelsWattenberg99], …, [DodisOstrovskyReyzinSmith04] …

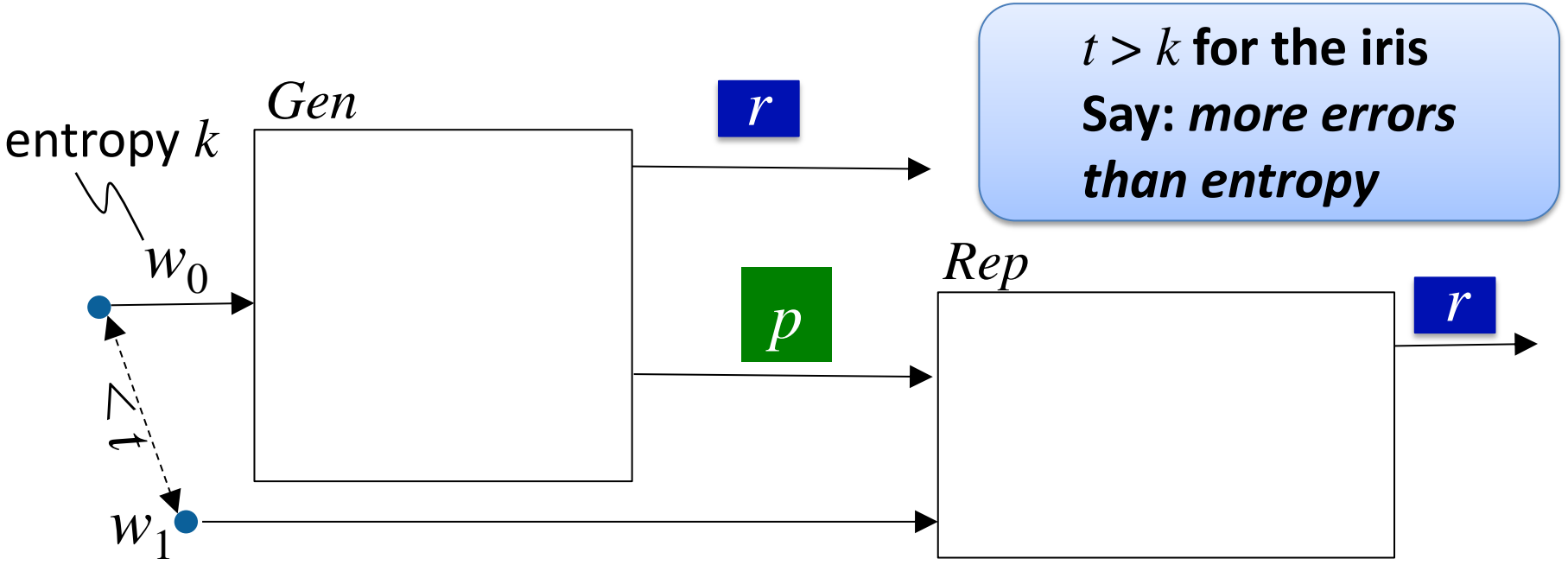
- Enrollment algorithm Gen:
  Take a measurement w_0 from the source.
  Use it to "lock up" random r in a nonsecret value p.
- Subsequent algorithm Rep: give same output if $d(w_0, w_1) < t$
- Security: r looks uniform even given p,
  when the source is good enough

Traditionally, security def. is information theoretic



Fuzzy Extractors: Goals

- Goal 1: handle as many sources as possible (typically, any source in which w_0 is 2^k-hard to guess)
- Goal 2: handle as much error as possible (typically, any w_1 within distance t)
- Most previous approaches are analyzed in terms of t and k
- Traditional approaches do not support sources with $t > k$

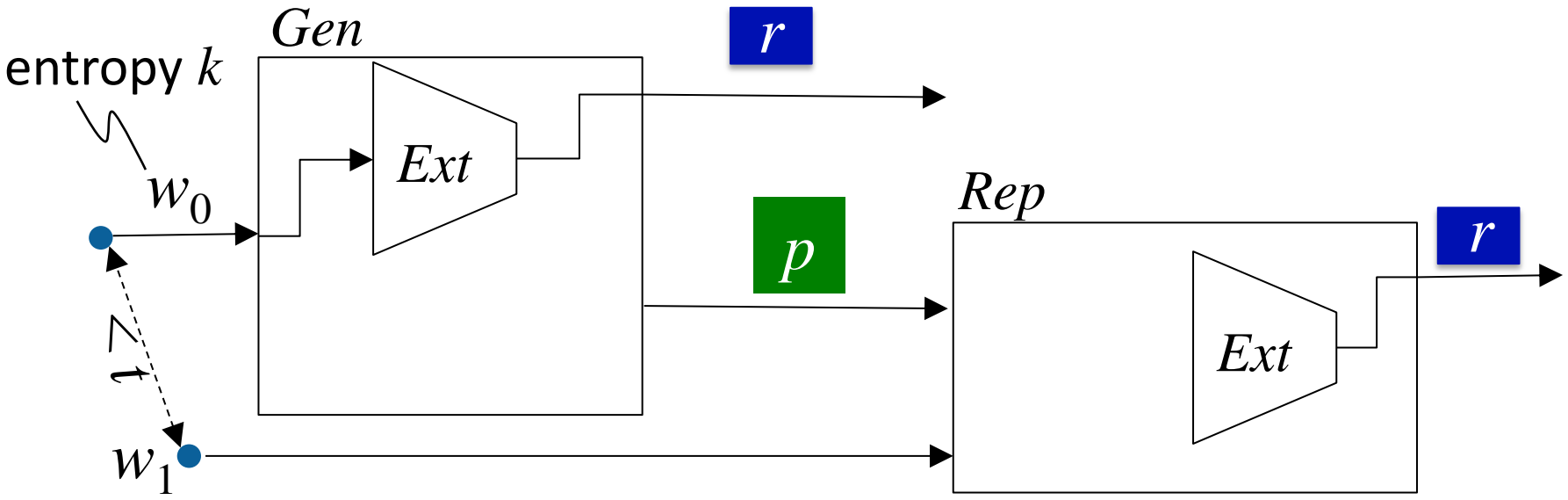


Contribution

- Lessons on how to construct fuzzy extractors when $t > k$ [FMR13,FRS14]
- First fuzzy extractors for large classes of distributions where $t > k$ [CFPRS14]
- First Reusable fuzzy extractor for arbitrary correlation between repeated readings [CFPRS14]
- Preliminary results on the iris

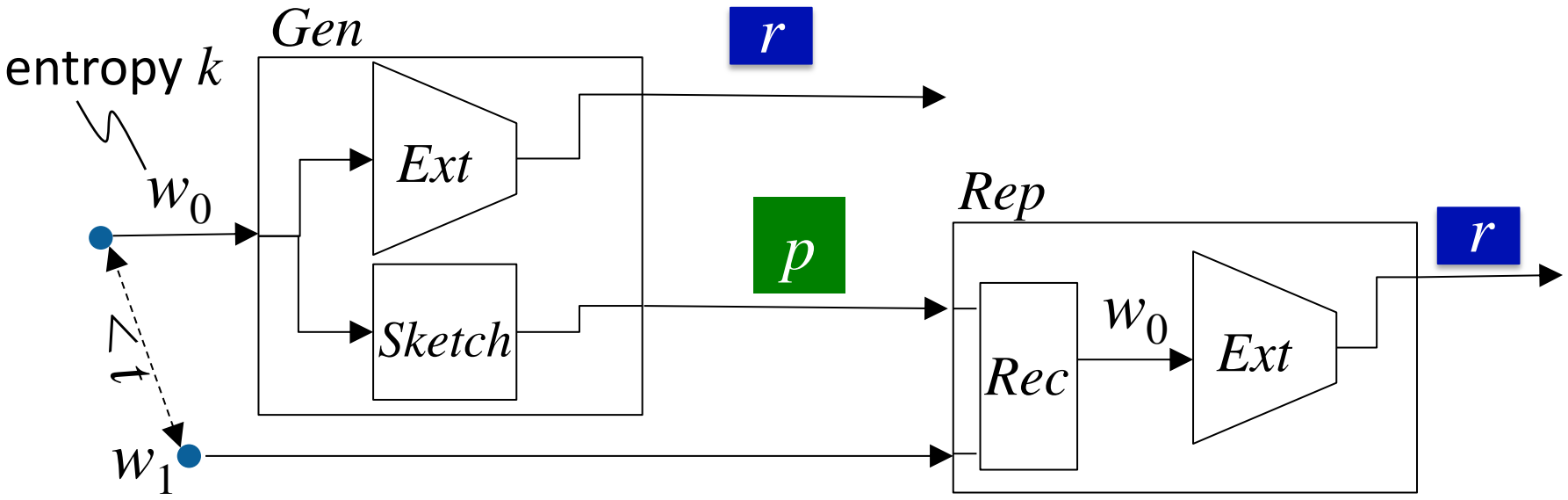
Fuzzy Extractors: Typical Construction

- derive r using a randomness extractor
  (converts high-entropy sources to uniform,
  e.g., via universal hashing [CarterWegman77])
- correct errors using a secure sketch [DodisOstrovskyReyzinSmith08]
  (gives recovery of the original from a noisy signal)

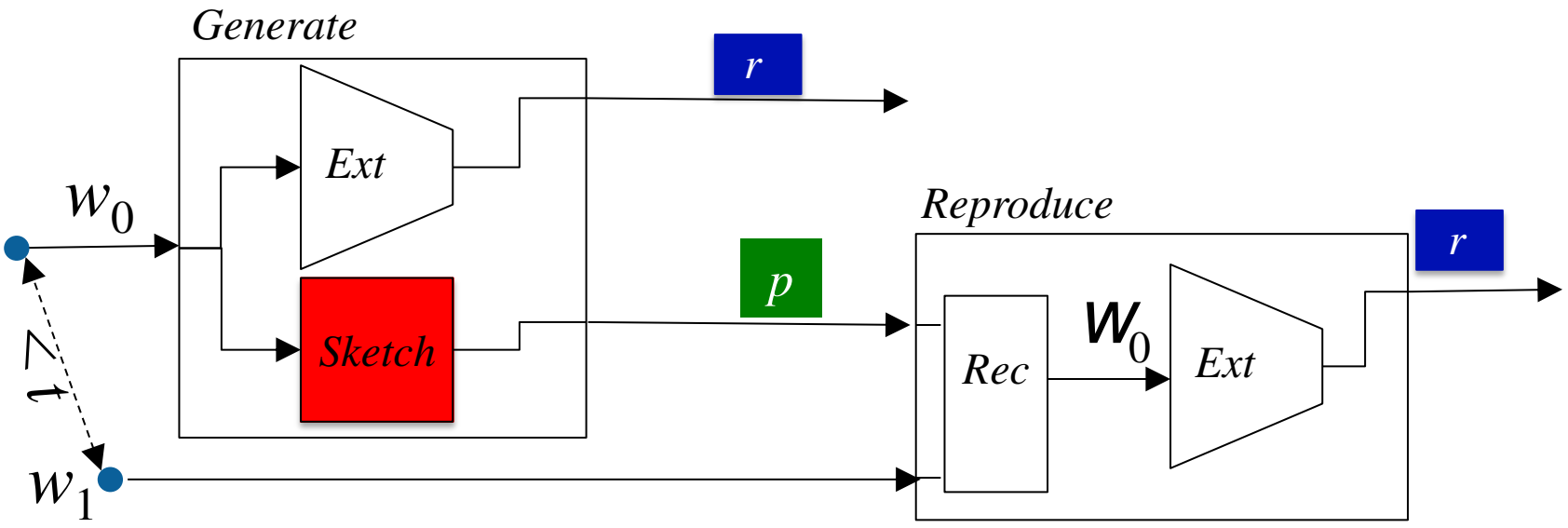


Fuzzy Extractors: Typical Construction

- derive r using a randomness extractor
  (converts high-entropy sources to uniform,
  e.g., via universal hashing [CarterWegman77])
- correct errors using a secure sketch [DodisOstrovskyReyzinSmith08]
  (gives recovery of the original from a noisy signal)

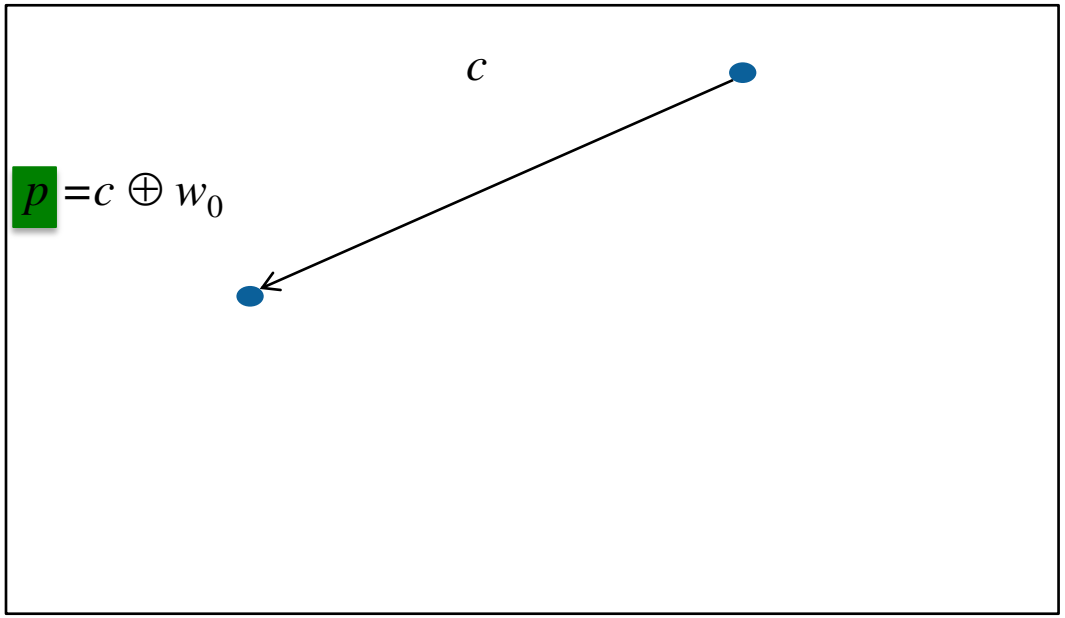


Secure Sketches

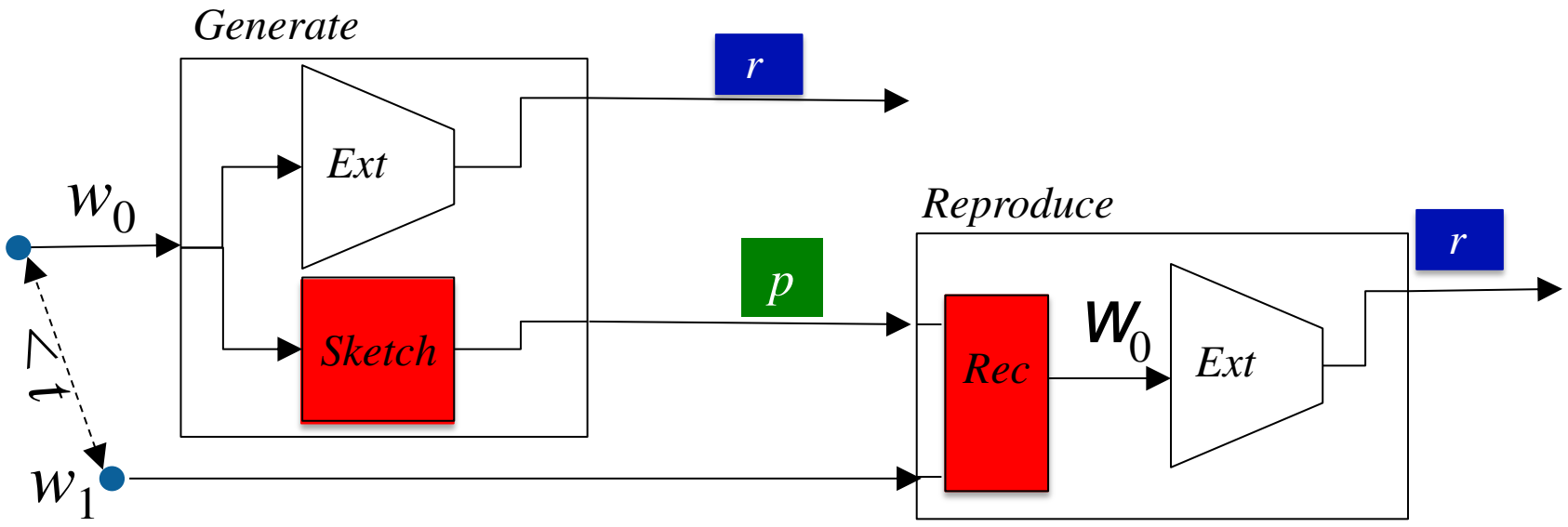


Code Offset Sketch
[JuelsWattenberg99]

C – Error correcting code correcting t errors

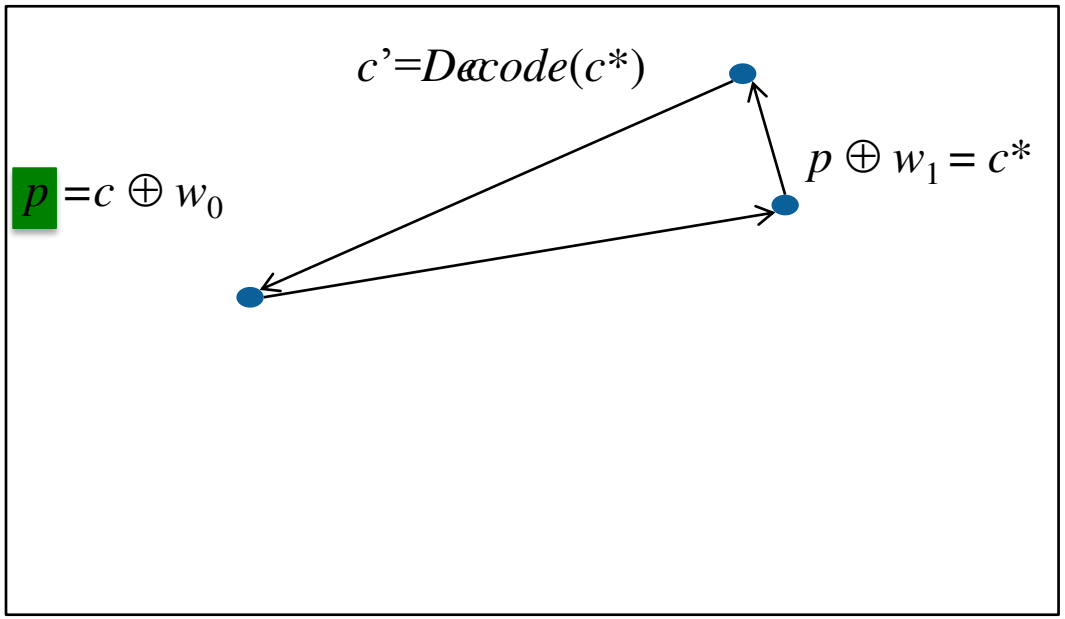


Secure Sketches



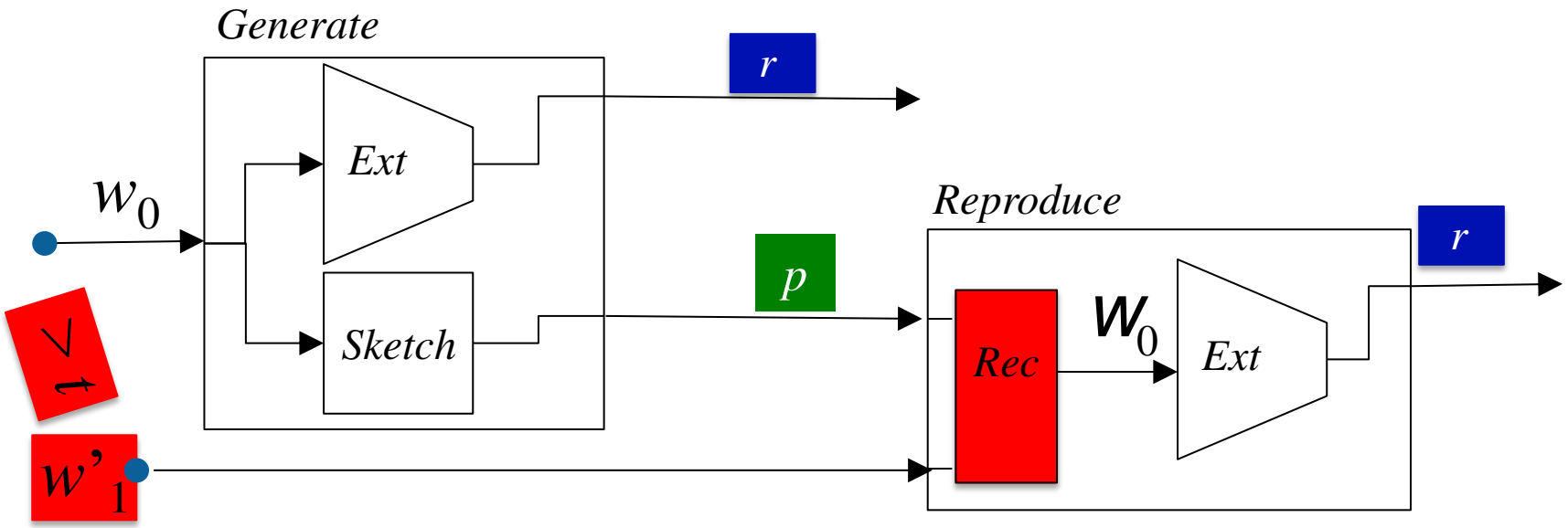
Code Offset Sketch
[JuelsWattenberg99]

C – Error correcting code correcting t errors



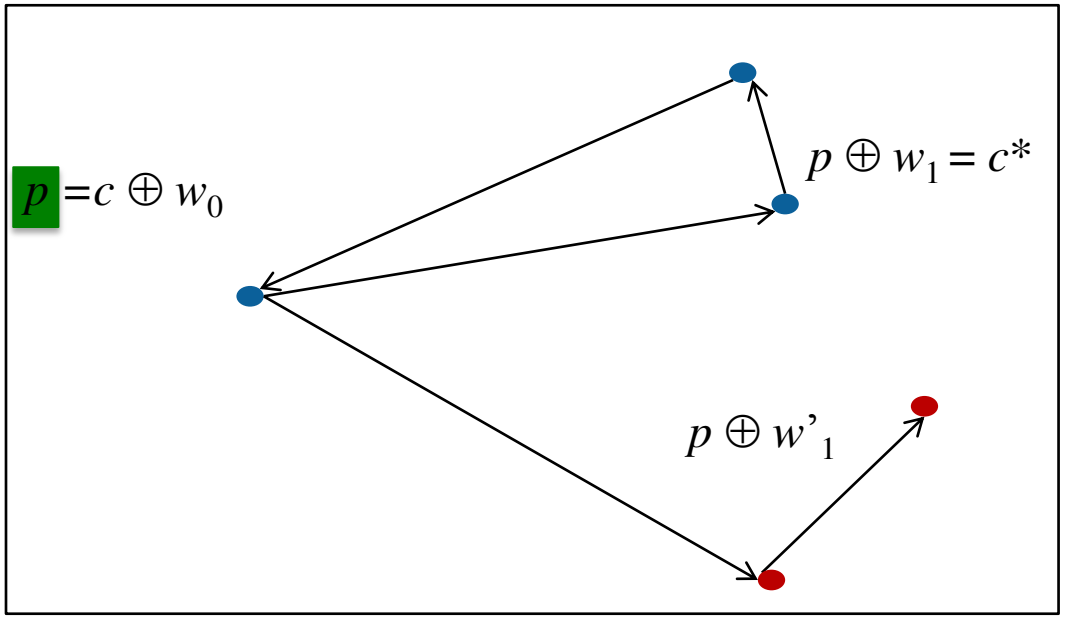
If decoding succeeds, $w_0 = c' \oplus p$.

Secure Sketches



Code Offset Sketch
[JuelsWattenberg99]

C – Error correcting code correcting t errors

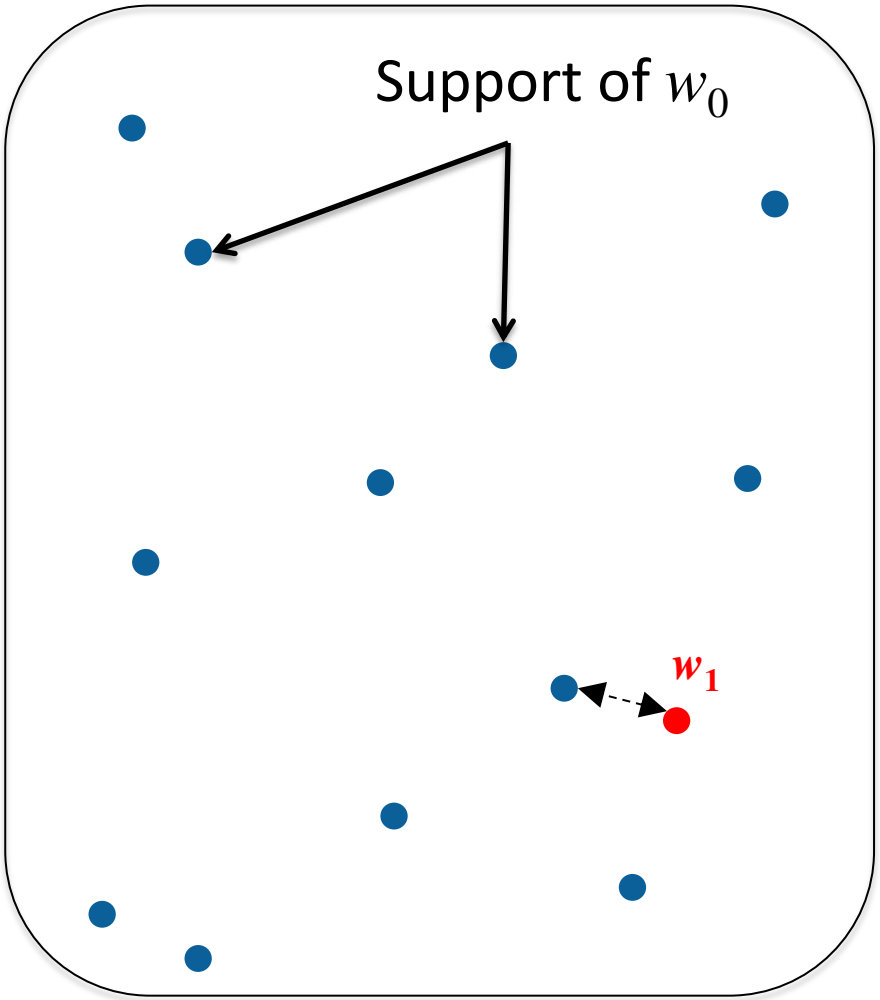


Goal: minimize how much p informs on w_0.

Outline

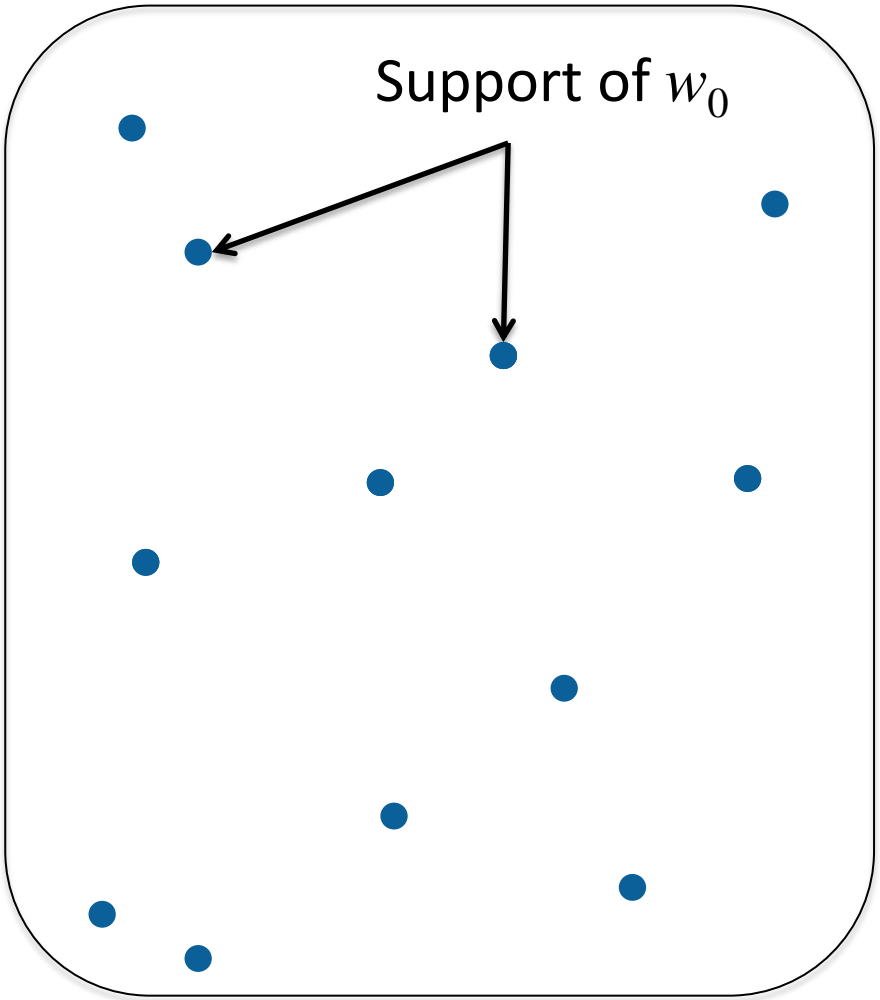
- Key Derivation from Noisy Sources
- Fuzzy Extractors
- Limitations of Traditional Approaches/Lessons
- New Constructions

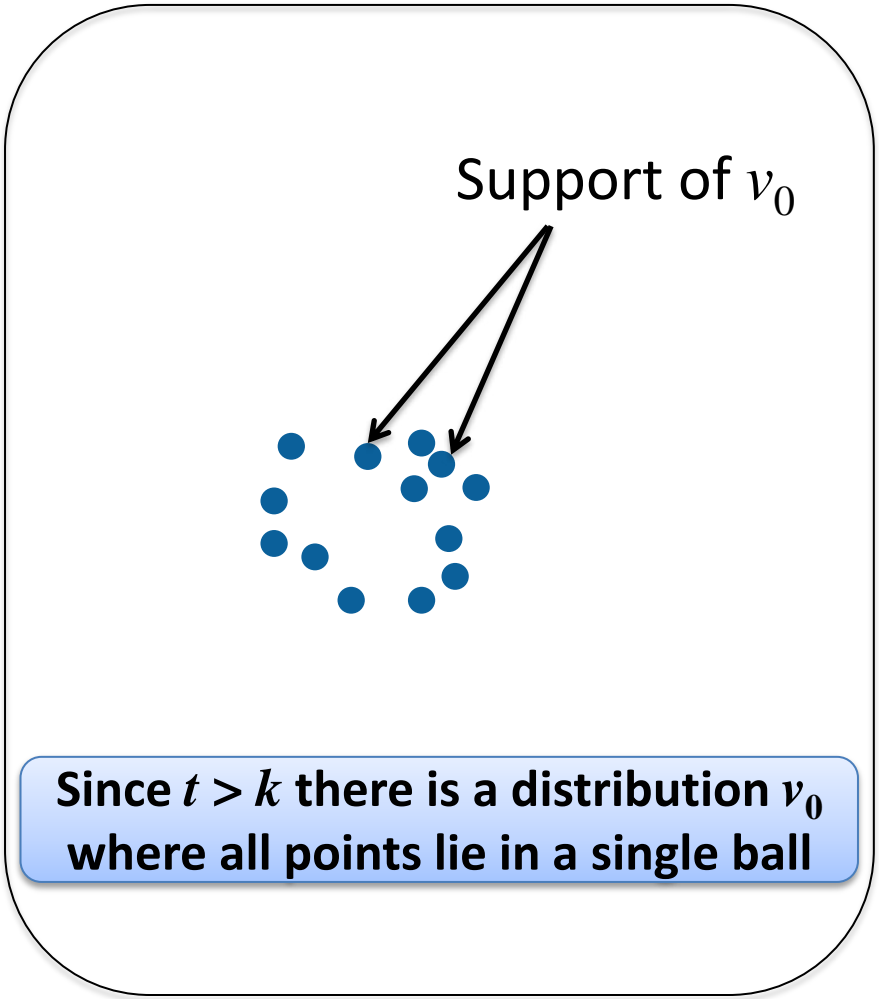
Is it possible to handle "more errors than entropy" ($t > k$)?



- This distribution has 2^k points
- Why might we hope to extract from this distribution?
- Points are far apart
  - No need to deconflict original reading

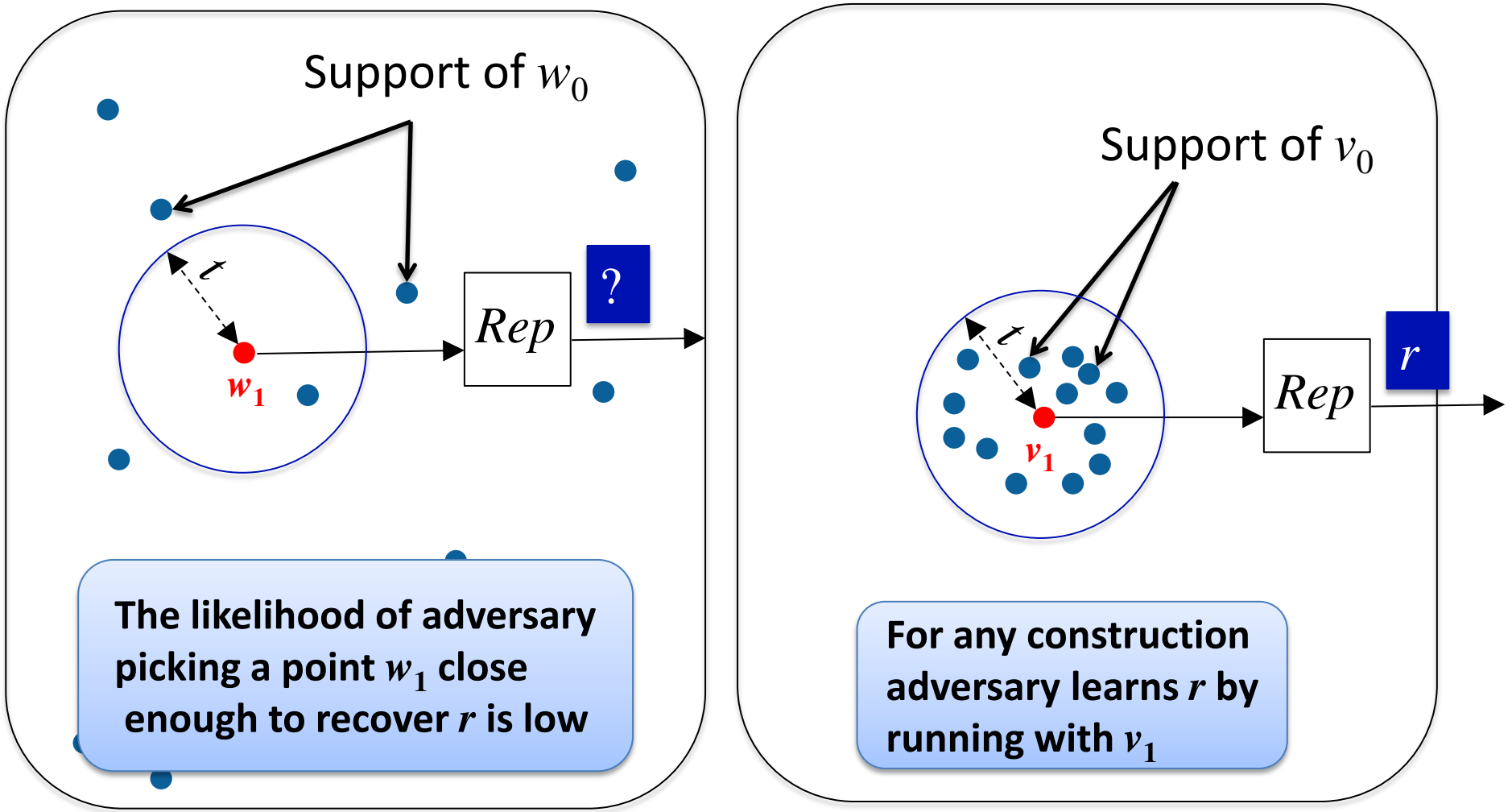
Is it possible to handle "more errors than entropy" ($t > k$)?



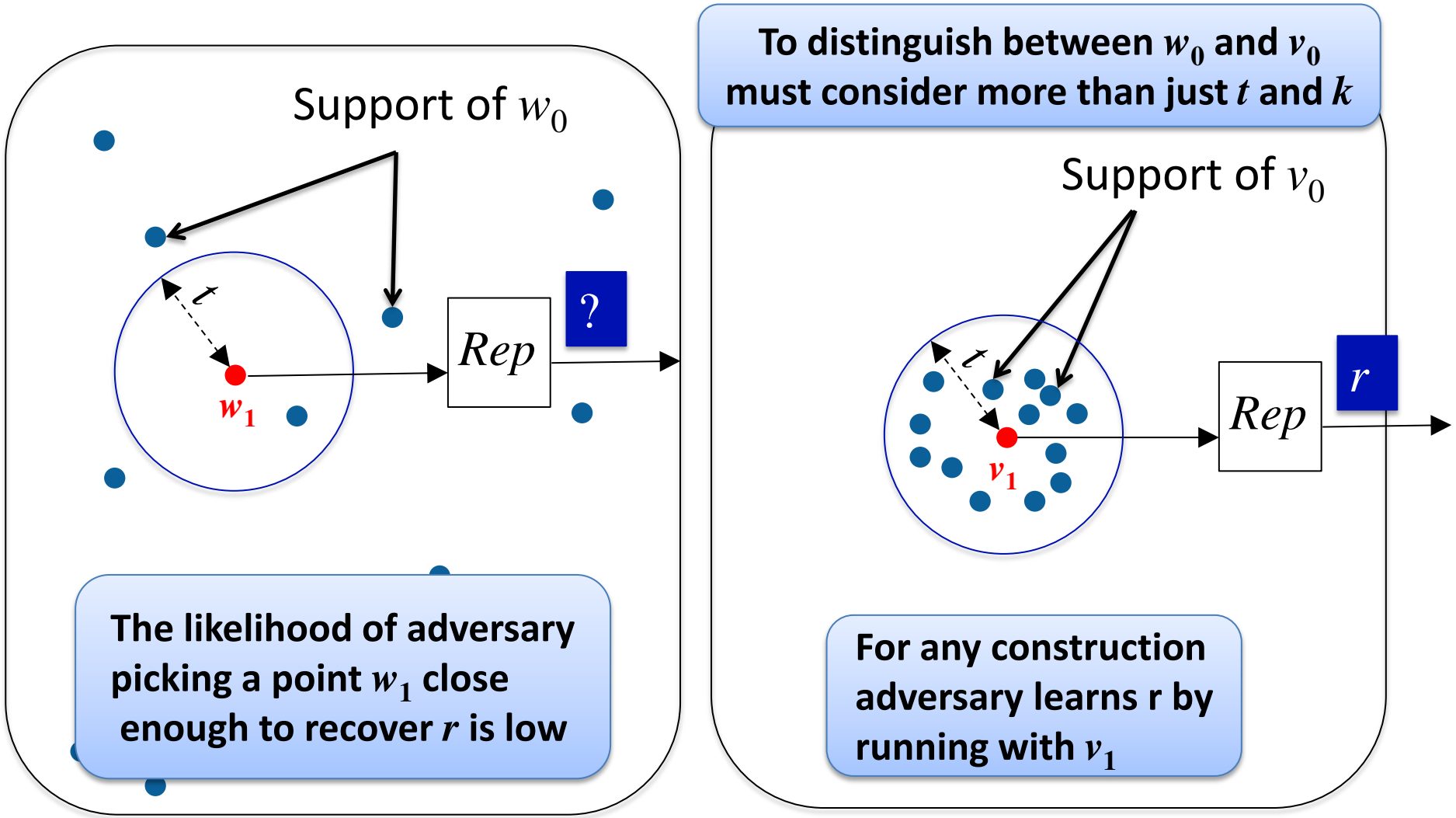


Left and right have same number of points and error tolerance

Is it possible to handle "more errors than entropy" ($t > k$)?



Is it possible to handle "more errors than entropy" ($t > k$)?

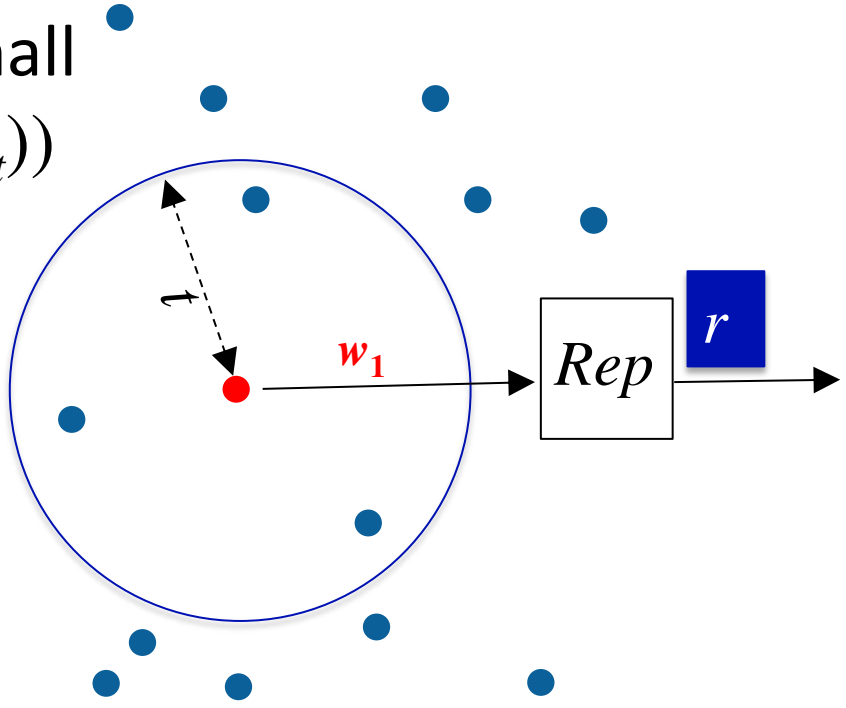


Lessons

1. Exploit structure of source beyond entropy
   - Need to understand what structure is helpful

Understand the structure of source

- Minimum necessary condition for fuzzy extraction: weight inside any B_t must be small
- Let $H_{\text{fuzz}}(W_0) = \log\,(1/\max\ \text{wt}(B_t))$
- Big $H_{\text{fuzz}}(W_0)$ is necessary
- Models security in ideal world
- Q: Is big $H_{\text{fuzz}}(W_0)$ sufficient for fuzzy extractors?



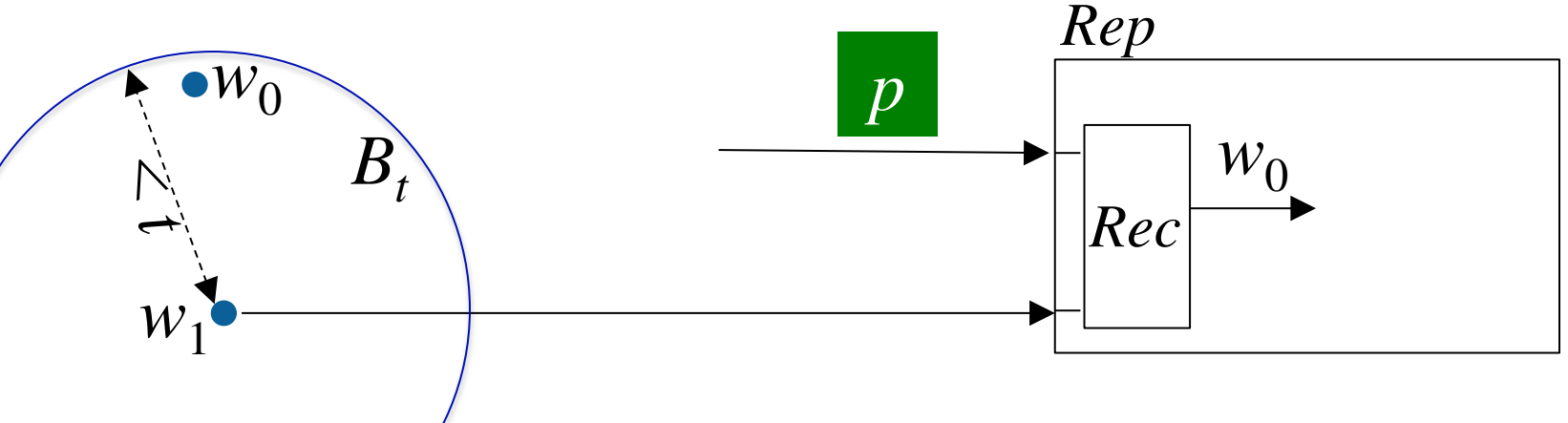
Is big $H_{\text{fuzz}}(W_0)$ sufficient?

- Thm [FRS]: Yes, if algorithms know exact distribution of W_0
- Imprudent to assume construction and adversary have same view of W_0
  - Should assume adversary knows more about W_0
  - Deal with adversary knowledge by providing security for family V of W_0, security should hold for whole family
- Thm [FRS]: No if W_0 is only known to come from a family V

Will show negative result for secure sketches (negative result for fuzzy extractors more complicated)

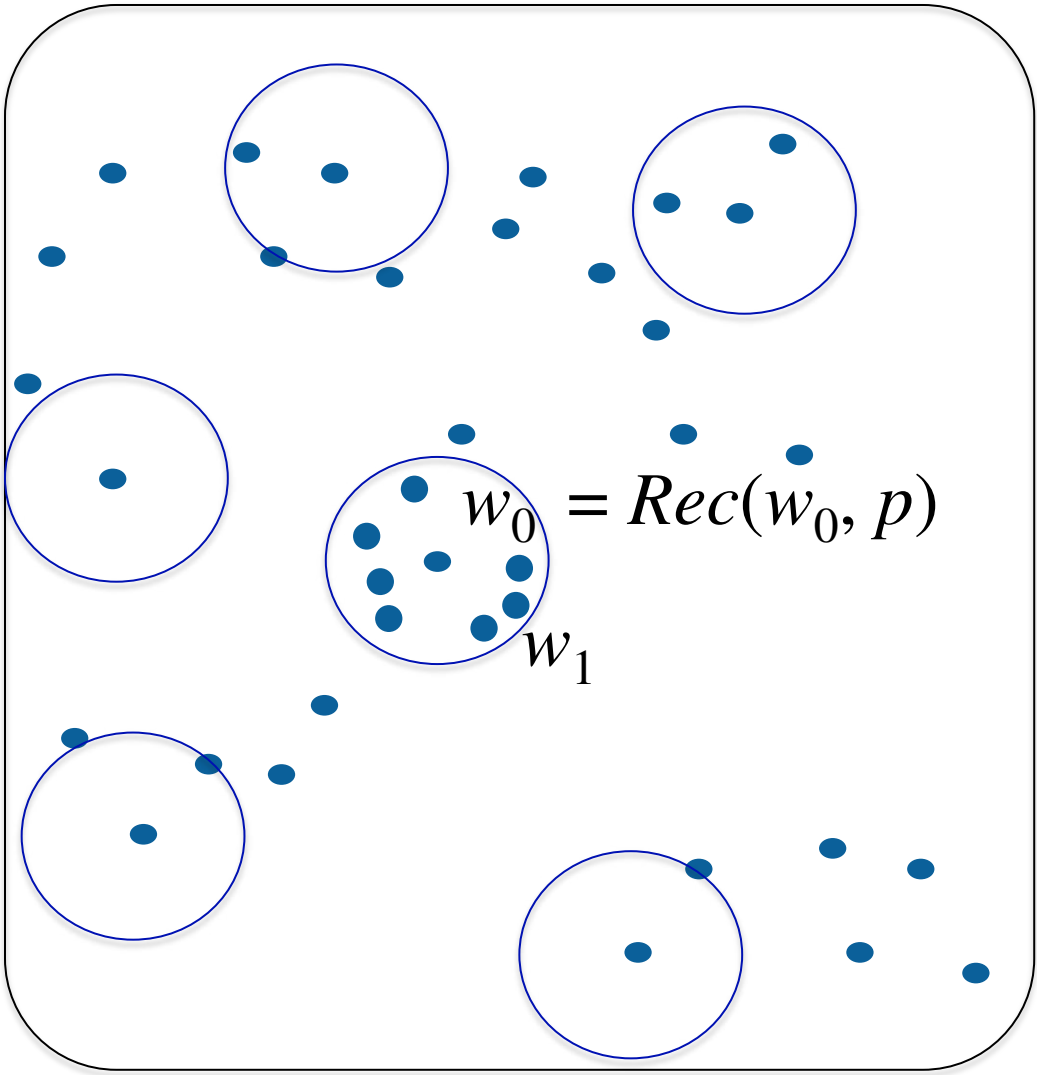
Thm [FRS]: No if W_0 comes from a family V

- Describe a family of distributions V
- For any secure sketch $Sketch$, Rec
  for most W_0 in V,
  few w^* in W_0 could produce p
- Implies W_0 has little entropy conditioned on p

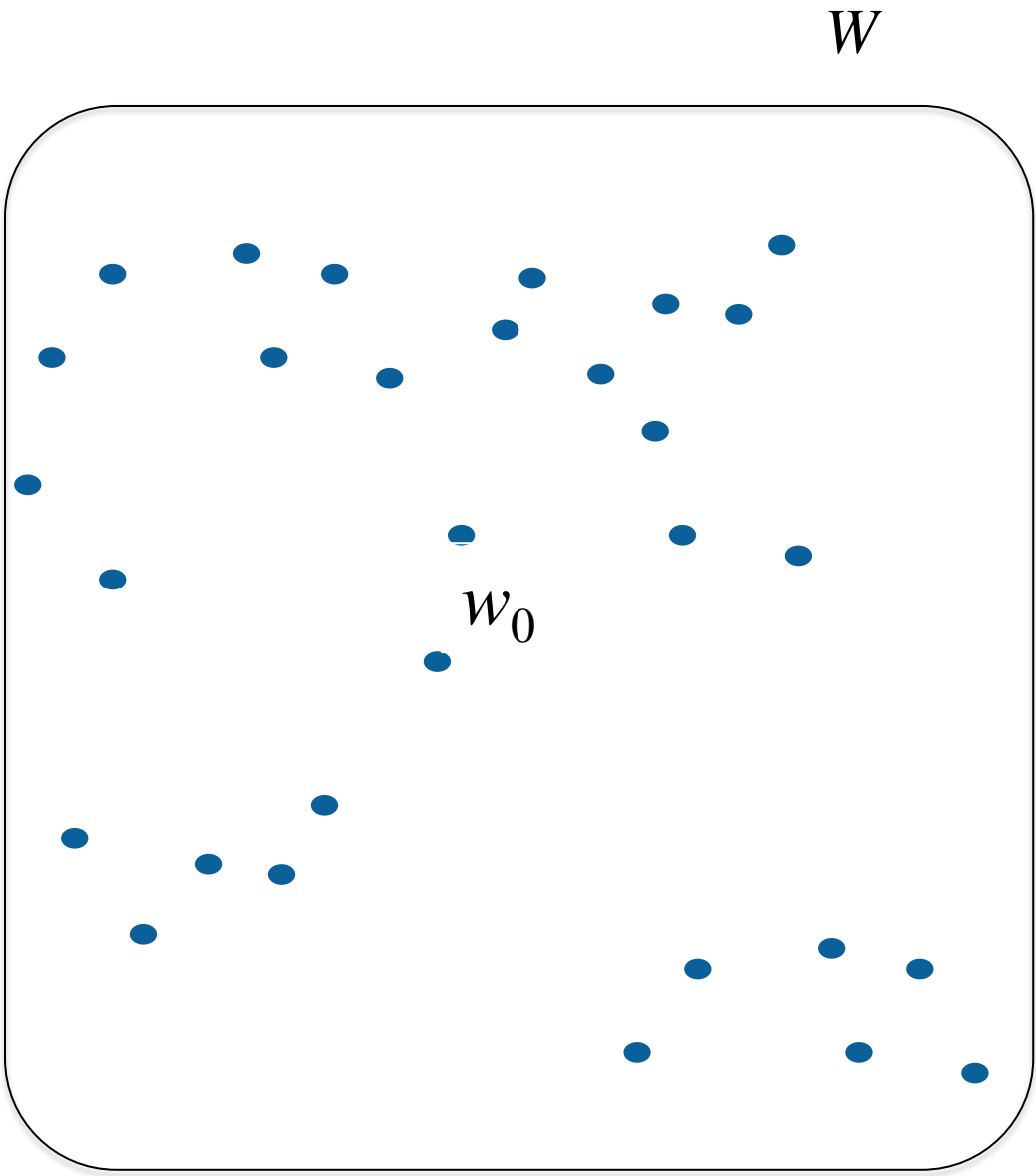


Now we'll consider family V, Adv. goal: most W in V, impossible to have many augmented fixed points

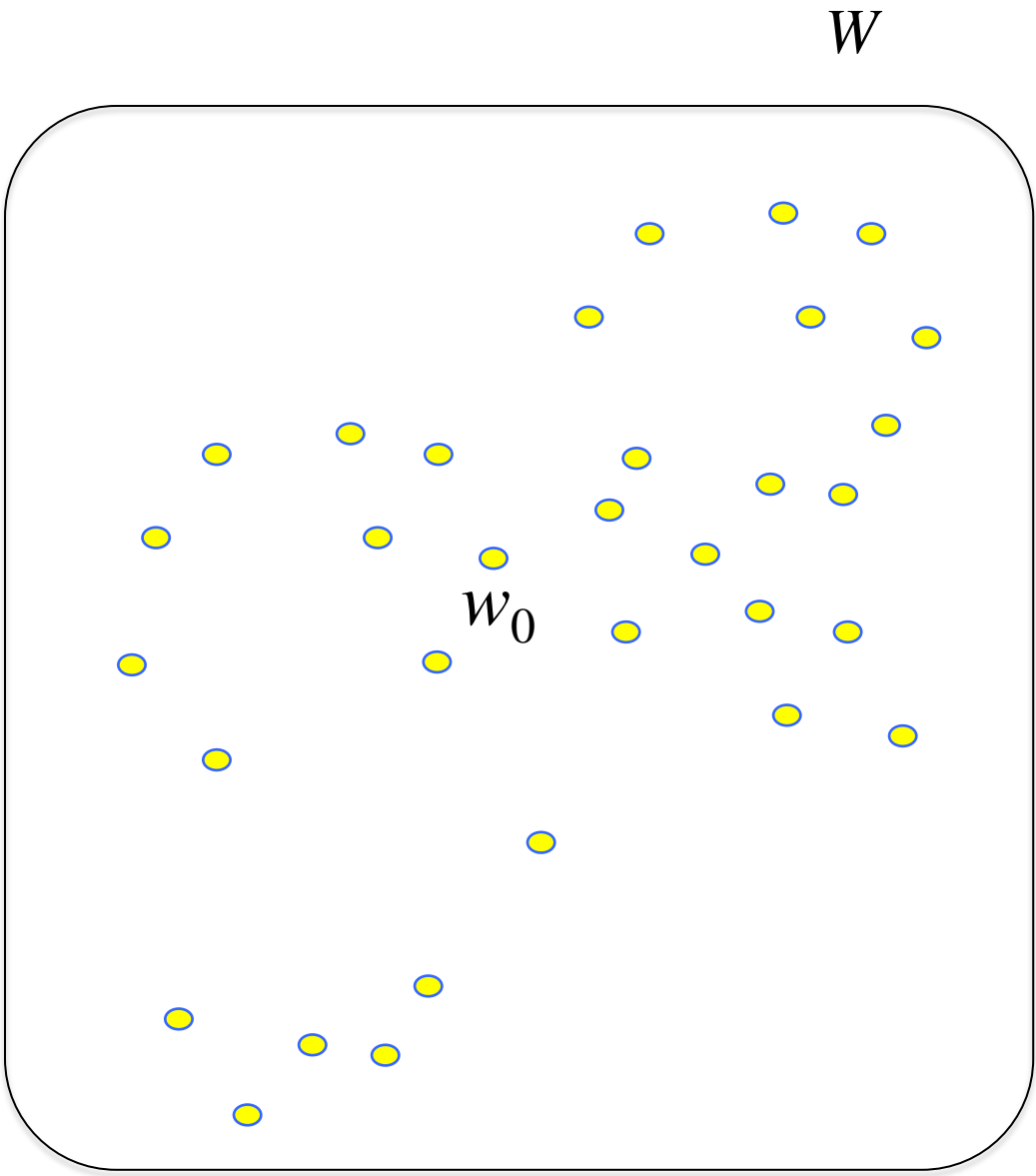
- Adversary specifies V
- Goal: build $Sketch$, Rec maximizing $H(W \mid p)$, for all W in V
- First consider one dist. W
- For w_0, $Rec(w_0, p) = w_0$
- For nearby w_1, $Rec(w_1, p) = w_0$
- Call augmented fixed point
- To maximize $H(W \mid p)$ make as many points of W augmented fixed points
- Augmented fixed points at least distance t apart (exponentially small fraction of space)



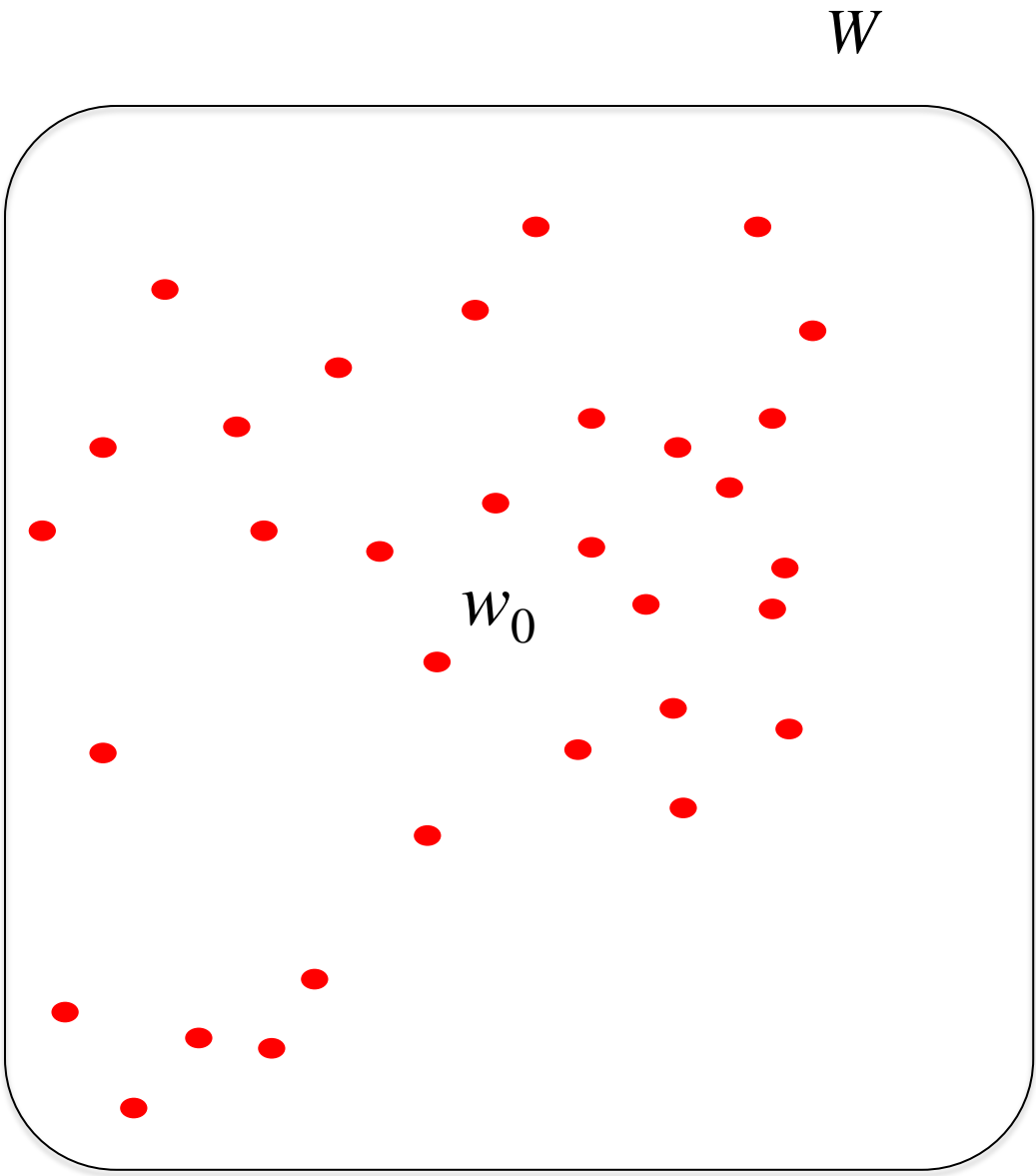
- Adversary specifies V
- Goal: build $Sketch$, Rec maximizing $H(W \mid p)$, for all W in V
- $Sketch$ must create augmented fixed points based only on w_0
- Build family with many possible distributions for each w_0
- $Sketch$ can't tell W from w_0



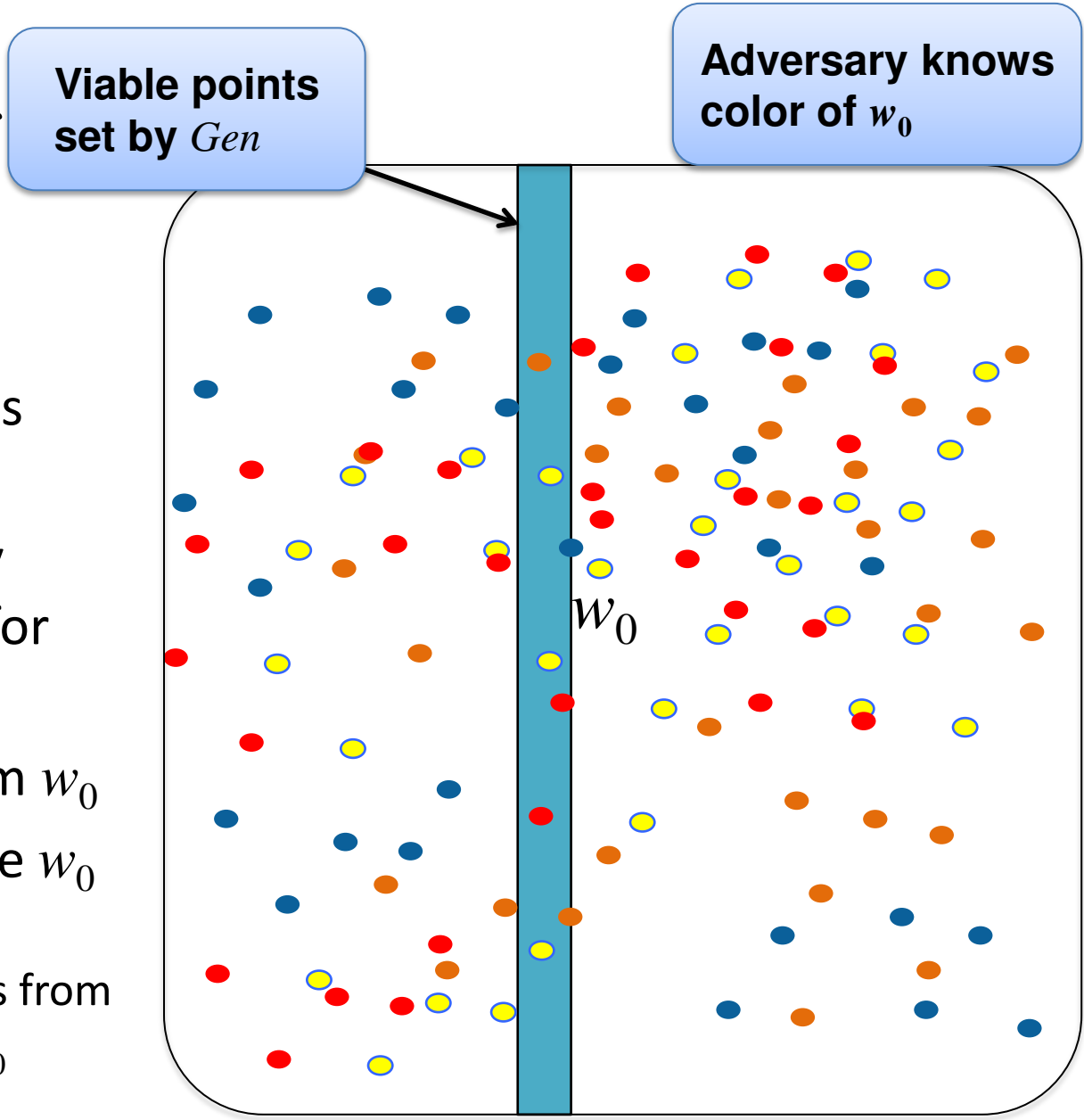
- Adversary specifies V
- Goal: build $Sketch$, Rec maximizing $H(W \mid p)$, for all W in V
- $Sketch$ must create augmented fixed points based only on w_0
- Build family with many possible distributions for each w_0
- $Sketch$ can't tell W from w_0



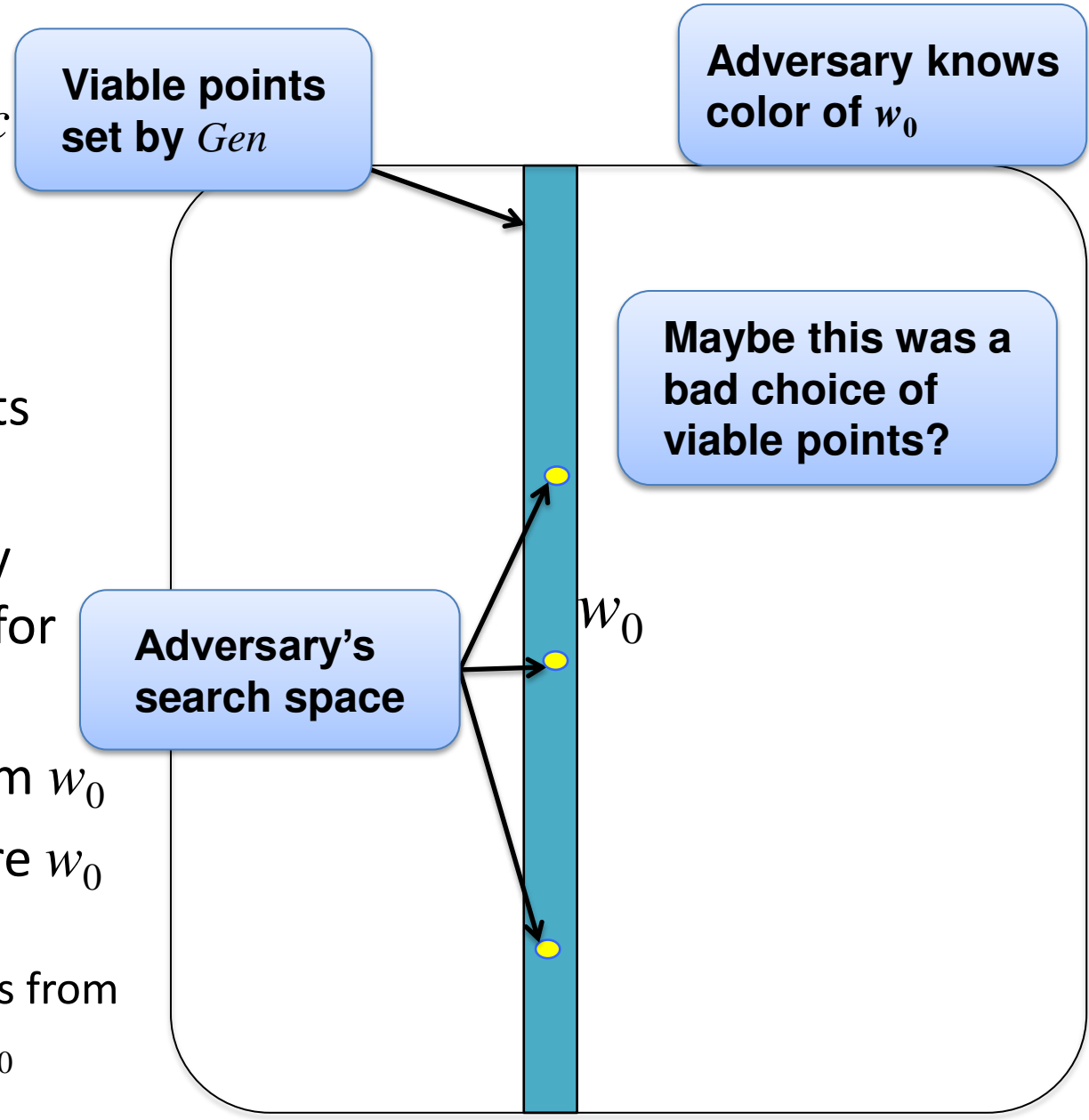
- Adversary specifies V
- Goal: build $Sketch$, Rec maximizing $H(W \mid p)$, for all W in V
- $Sketch$ must create augmented fixed points based only on w_0
- Build family with many possible distributions for each w_0
- $Sketch$ can't tell W from w_0



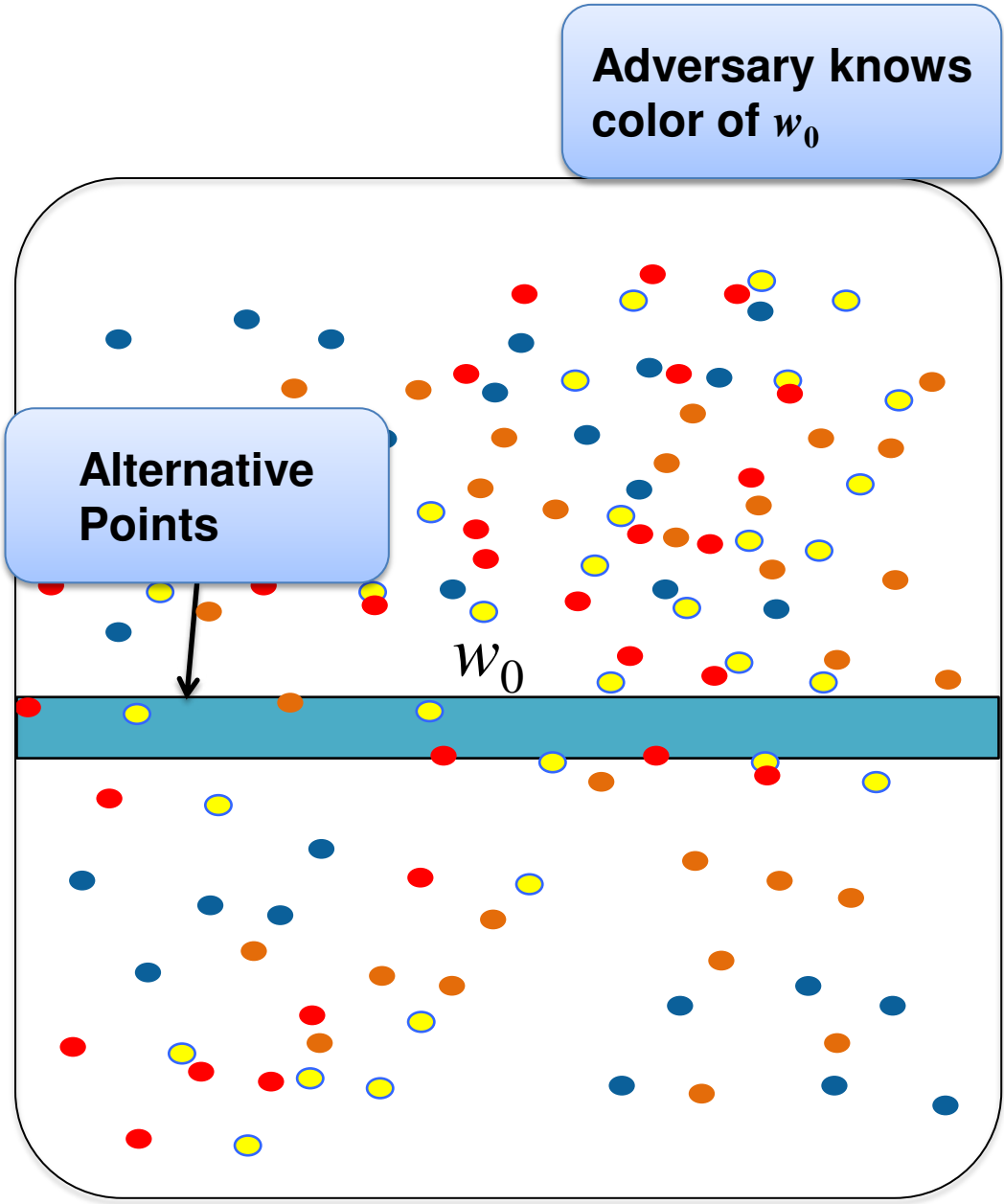
- Adversary specifies V
- Goal: build $Sketch$, Rec maximizing $H(W \mid p)$, for all W in V
- $Sketch$ must create augmented fixed points based only on w_0
- Build family with many possible distributions for each w_0
- $Sketch$ can't tell W from w_0
- Distributions only share w_0
  - $Sketch$ must include augmented fixed points from all distributions with w_0



- Adversary specifies V
- Goal: build $Sketch$, Rec maximizing $H(W \mid p)$, for all W in V
- $Sketch$ must create augmented fixed points based only on w_0
- Build family with many possible distributions for each w_0
- $Sketch$ can't tell W from w_0
- Distributions only share w_0
  - $Sketch$ must include augmented fixed points from all distributions with w_0



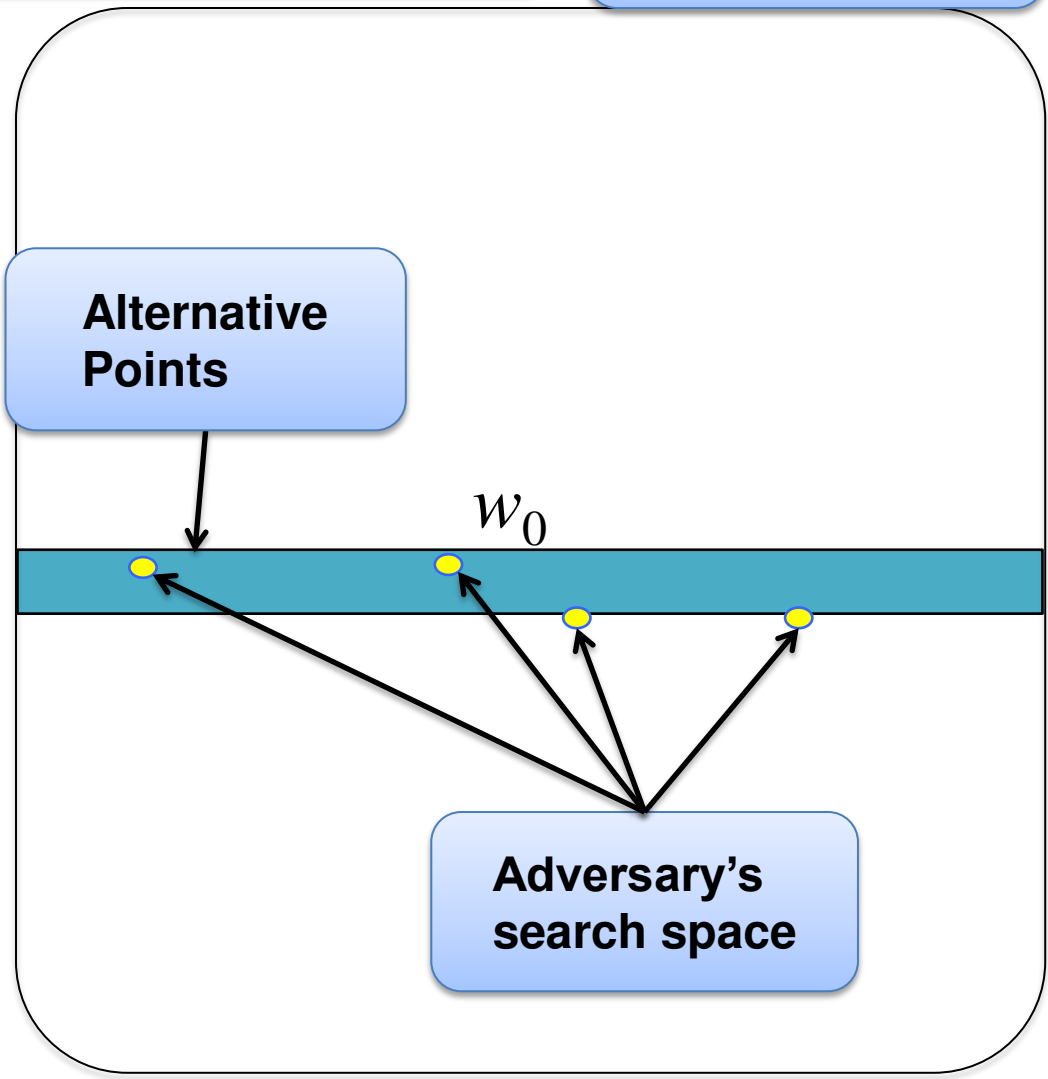
- Adversary specifies V
- Goal: build $Sketch$, Rec maximizing $H(W \mid p)$, for all W in V
- $Sketch$ must create augmented fixed points based only on w_0
- Build family with many possible distributions for each w_0
- $Sketch$ can't tell W from w_0
- Distributions only share w_0
  - $Sketch$ must include augmented fixed points from all distributions with w_0



- Adversary specifies V
- Goal: build $Sketch$, Rec maximizing $H(W \mid p)$, for all W in V
- $Sketch$ must create augmented fixed points based only on w_0
- Build family with many possible distributions for each w_0
- $Sketch$ can't tell W from w_0
- Distributions only share w_0
  - $Sketch$ must include augmented fixed points from all distributions with w_0

Thm: $Sketch$, Rec **can include at most** 4 **augmented fixed points from members of** V **on average**

Adversary knows color of w_0



Is big $H_{\text{fuzz}}(W_0)$ sufficient?

- Thm [FRS]: Yes, if algorithms know exact distribution of W_0
- Imprudent to assume construction and adversary have same view of W_0
  - Deal with adversary knowledge by providing security for family V of W_0, security should hold for whole family
- Thm [FRS]: No if adversary knows more about W_0 than fuzzy extractor creator

Fuzzy extractors defined information-theoretically (used info-theory tools),
No compelling need for info-theory security

Lessons

1. Stop using secure sketches

2. Define objects computationally

Thm [FMR13]: Natural definition of computational secure sketches (pseudo entropy) limited:

Can build sketches with info-theoretic security from sketches that provide computational security

3. Stop using secure sketches

Outline

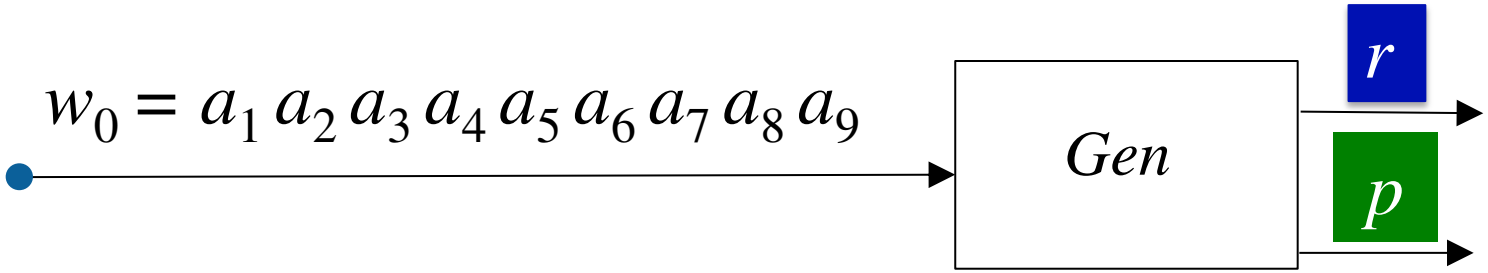
- Key Derivation from Noisy Sources
- Traditional Fuzzy Extractors
- Lessons
  1. Exploit structure of source beyond entropy
  2. Define objects computationally
  3. Stop using secure sketches

New Constructions

Idea [CFPRS14]: "encrypt" r using parts of w_0

Gen: - get random combinations of symbols in w_0

- "lock" r using these combinations

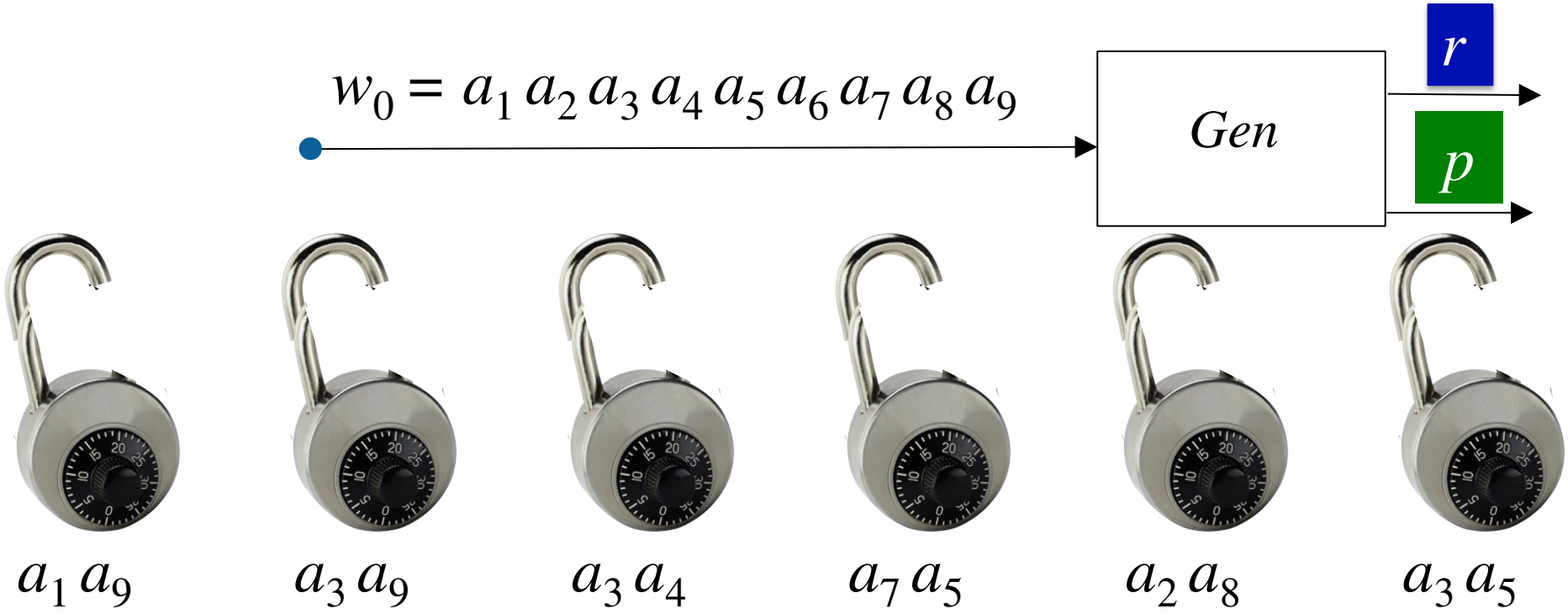


$a_1\, a_9$ $a_3\, a_9$ $a_3\, a_4$ $a_7\, a_5$ $a_2\, a_8$ $a_3\, a_5$

Idea [CFPRS14]: "encrypt" r using parts of w_0

Gen: - get random combinations of symbols in w_0

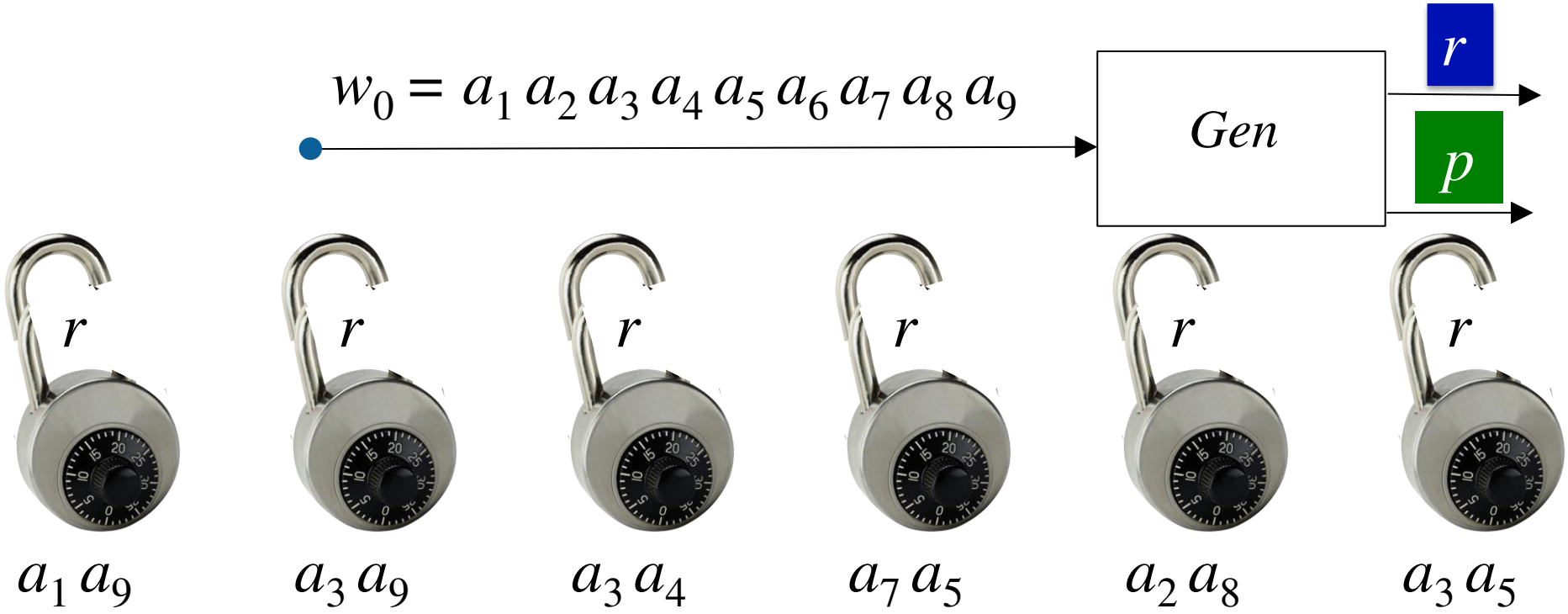
- "lock" r using these combinations



Idea [CFPRS14]: "encrypt" r using parts of w_0

Gen: - get random combinations of symbols in w_0

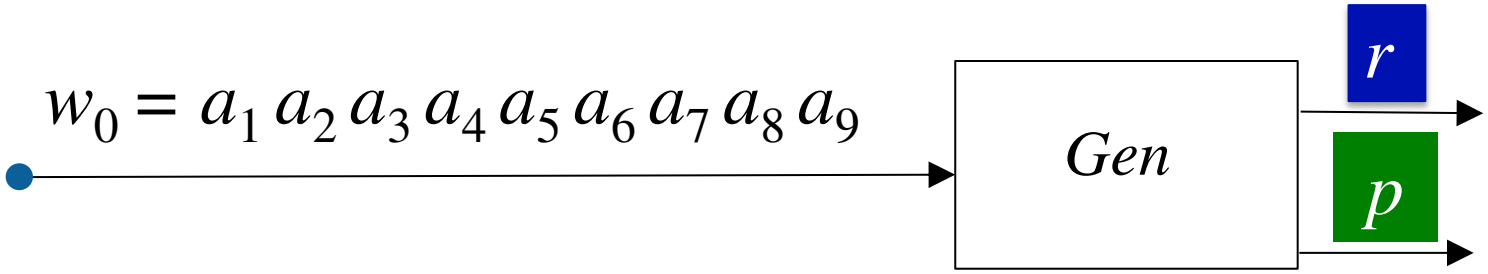
- "lock" r using these combinations



Idea [CFPRS14]: "encrypt" r using parts of w_0

Gen: - get random combinations of symbols in w_0

- "lock" r using these combinations
- p = locks + positions of symbols needed to unlock

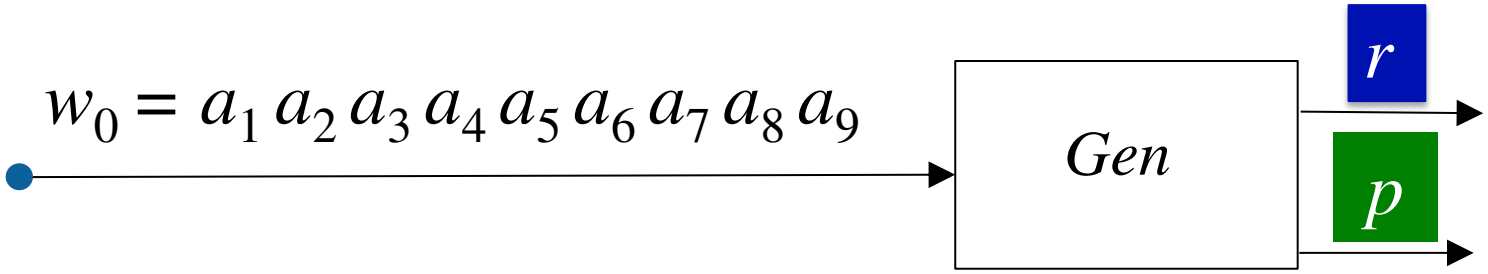


$a_1\, a_9$ $a_3\, a_9$ $a_3\, a_4$ $a_7\, a_5$ $a_2\, a_8$ $a_3\, a_5$

Idea [CFPRS14]: "encrypt" r using parts of w_0

Gen: - get random combinations of symbols in w_0

- "lock" r using these combinations
- p = locks + positions of symbols needed to unlock



1 9 3 9 3 4 7 5 2 8 3 5

Idea [CFPRS14]: "encrypt" r using parts of w_0

Gen: - get random combinations of symbols in w_0

- "lock" r using these combinations
- p = locks + positions of symbols needed to unlock

p

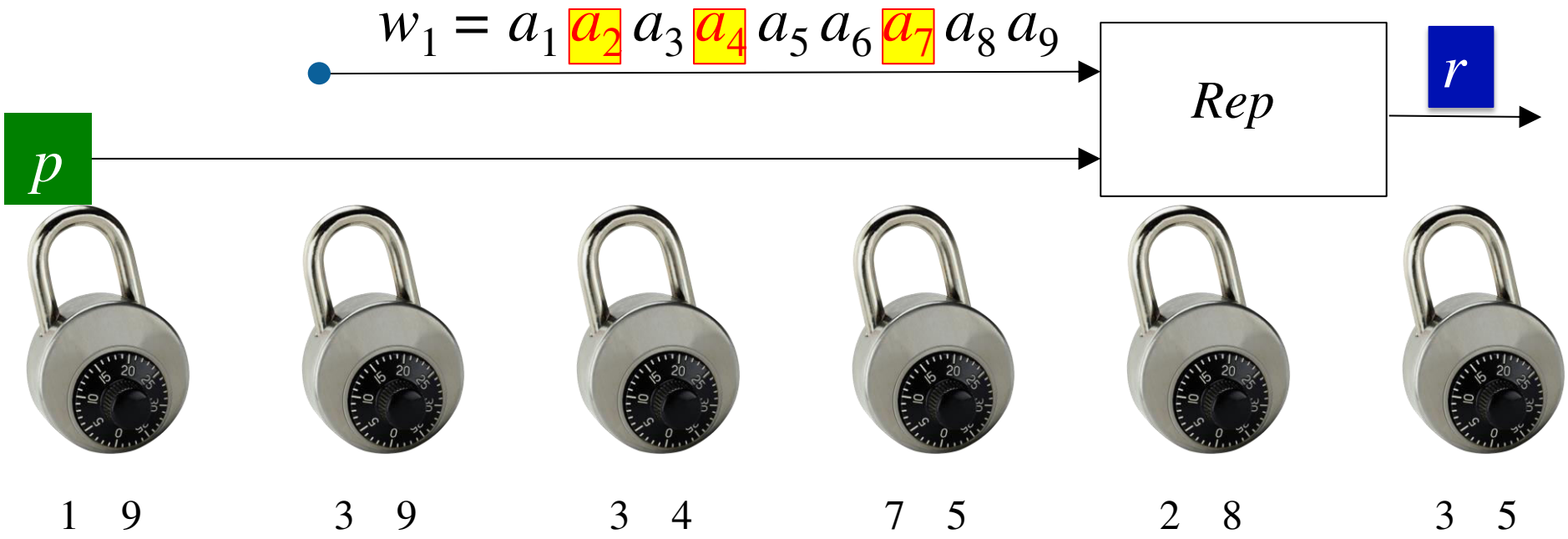
1 9 3 9 3 4 7 5 2 8 3 5

Idea [CFPRS14]: "encrypt" r using parts of w_0

Gen: - get random combinations of symbols in w_0

- "lock" r using these combinations
- p = locks + positions of symbols needed to unlock

Rep:

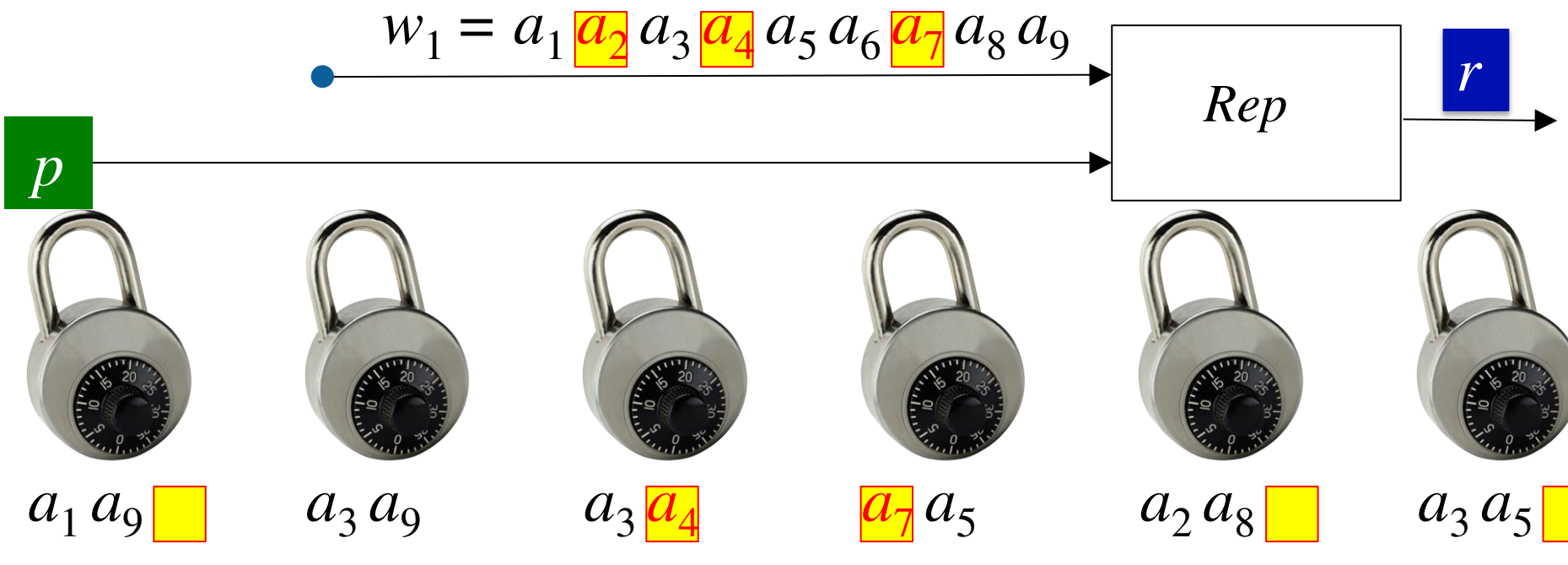


Idea [CFPRS14]: "encrypt" r using parts of w_0

Gen: - get random combinations of symbols in w_0

- "lock" r using these combinations
- p = locks + positions of symbols needed to unlock

Rep:

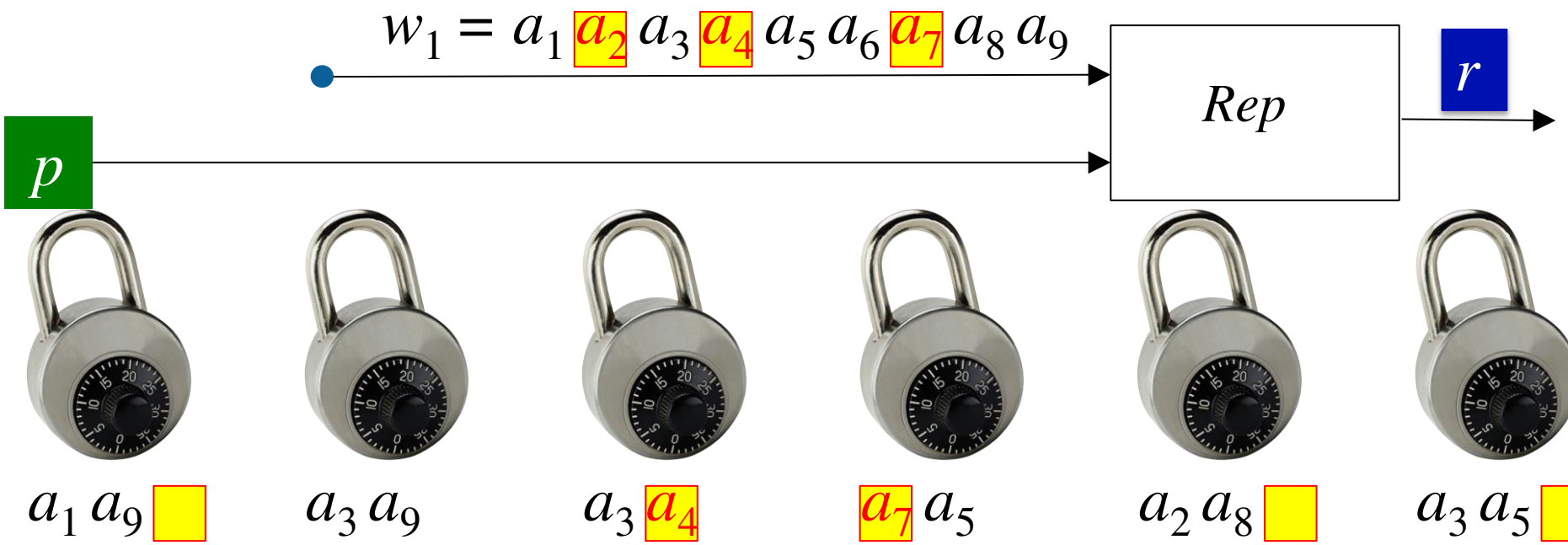


Idea [CFPRS14]: "encrypt" r using parts of w_0

Gen: - get random combinations of symbols in w_0

- "lock" r using these combinations
- p = locks + positions of symbols needed to unlock

Rep: Use the symbols of w_1 to open at least one lock

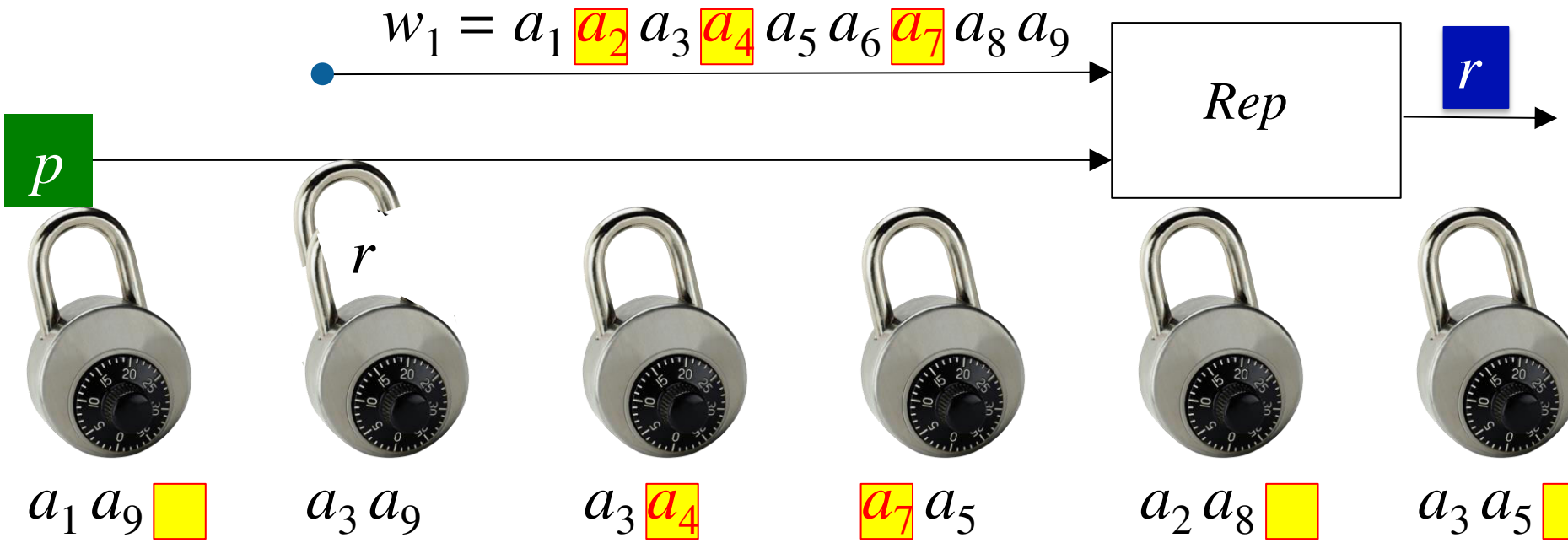


Idea [CFPRS14]: "encrypt" r using parts of w_0

Gen: - get random combinations of symbols in w_0

- "lock" r using these combinations
- p = locks + positions of symbols needed to unlock

Rep: Use the symbols of w_1 to open at least one lock



Idea [CFPRS14]: "encrypt" r using parts of w_0

Gen: - get random combinations of symbols in w_0

- "lock" r using these combinations
- p = locks + positions of symbols needed to unlock

Rep: Use the symbols of w_1 to open at least one lock

Error-tolerance:

one combination must unlock with high probability

Security: each combination must have enough entropy

(sampling of symbols must preserve sufficient entropy)

How to implement locks?

- A lock is the following program:
  - If input = $a_1\, a_9\, a_2$, output r
  - Else output $\perp$
  - One implementation (R.O. model): lock = $r \oplus \mathrm{H}(a_1\, a_9\, a_2)$

$a_1\, a_9$

a_2

- Ideally: Obfuscate this program
  - Obfuscation: preserve functionality, hide the program
  - Obfuscating this specific program called “digital locker”

Digital Lockers

$a_1\, a_9\, a_2$

- Digital Locker is obfuscation of
  - If input = $a_1\, a_9\, a_2$, output r
  - Else output $\perp$
- Equivalent to encryption of r that is secure even multiple times with correlated, weak keys [CanettiKalaiVariaWichs10]
- Digital lockers are practical (R.O. or DL-based) [CanettiDakdouk08], [BitanskyCanetti10]
- Hides r if input can't be exhaustively searched (superlogarithmic entropy)

Digital Lockers

$a_1\, a_9$

a_2

- Digital Locker is obfuscation of
  - If input = $a_1\, a_9\, a_2$, output r
  - Else output $\perp$

- Q: if you are going to use obfuscation, why bother? Why not just obfuscate the following program for p
  - If distance between w_0 and the input is less than t, output r
  - Else output $\perp$

- A: you can do that [BitanskyCanettiKalaiPaneth14], except it's very impractical + has a very strong assumption

How good is this construction?

- Handles sources with $t > k$
- For correctness: $t <$ constant fraction of symbols

$a_1\, a_9$

$a_3\, a_9$

$a_3\, a_4$

$a_7\, a_5$

$a_2\, a_8$

$a_3\, a_5$

How good is this construction?

- Handles sources with $t > k$
- For correctness: $t <$ constant fraction of symbols

Construction 2:

Supports $t =$ constant fraction but only for really large alphabets

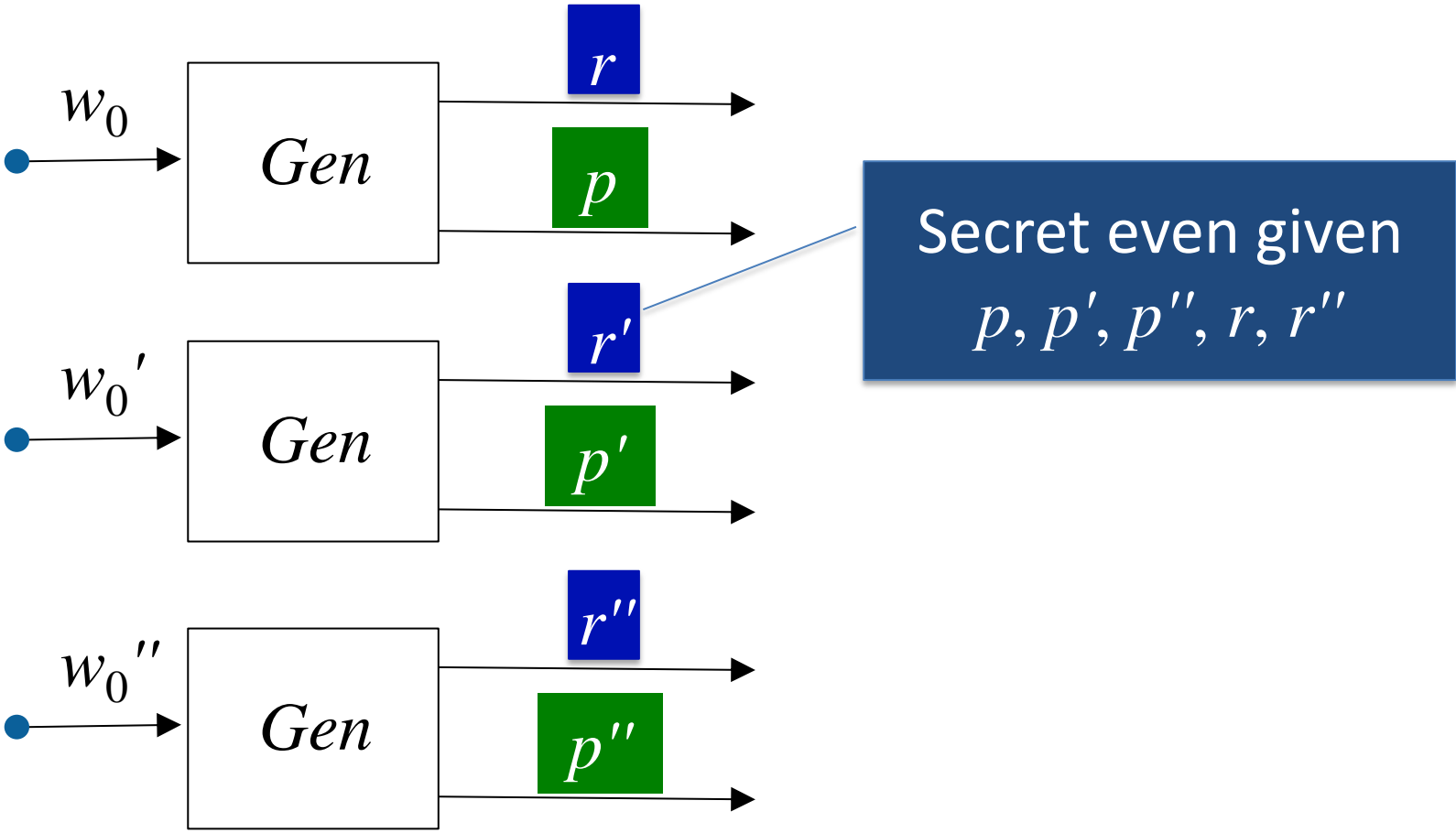
Construction 3:

Similar parameters but info-theoretic security

Why did I tell you about computation constructional?

How good is this construction?

- It is reusable!
  - Same source can be enrolled multiple times with multiple independent services



How good is this construction?

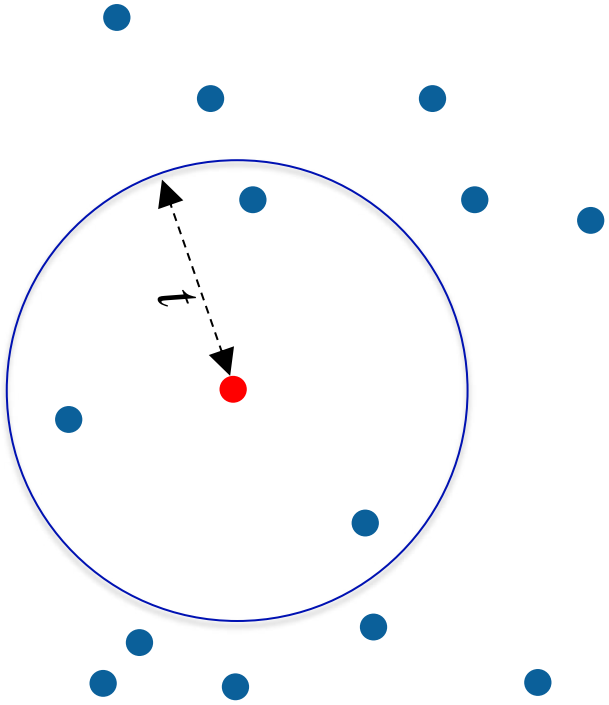
- It is reusable!
  - Same source can be enrolled multiple times with multiple independent services
  - Follows from composability of obfuscation
  - In the past: difficult to achieve, because typically new enrollments leak fresh information
  - Only previous construction [Boyen2004]: all reading must differ by fixed constants (unrealistic)
  - Our construction: each reading individually must satisfy our conditions

How good is this construction?

- It is reusable!
- Looks promising for the iris
  - Security: need samples of iris code bits are high entropy
    - First look: 100 bit sample of iris code has 60 bits of entropy
  - Correctness: unlock at one lock with high probability
    - Fuzzy extractors for iris codes should support 10% errors for high probability of recovering key
    - Takes $170{,}000$ combination locks on 100 bit input (impractical for client server, feasible for personal devices)
  - Next step: verify irises satisfy properties needed for security of construction

Conclusion

- Lessons:
  - Exploit structure in source
  - Provide computational security
  - Don't use secure sketches (i.e., full error correction)



- It is possible to cover sources with more errors than entropy!
- Also get reusability!
- Preliminary iris results promising

Questions?