Generating Strong Keys from Noisy Data

Yevgeniy Dodis, New York U

Leo Reyzin, Boston U

Adam Smith, MIT

Using Noisy Data for Passwords

- Crypto moving beyond encryption / authentication
- Newer applications often poorly modeled
  - Ad hoc solutions
  - Get broken!
- Crypto Theory: models, proofs
- **This talk**: – formal framework

  – provably secure constructions

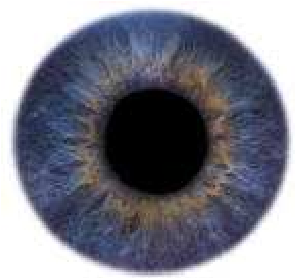
for using biometric data for authentication, key recovery

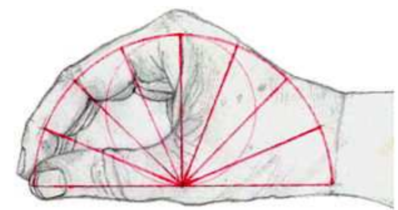
The Problem: We're Human

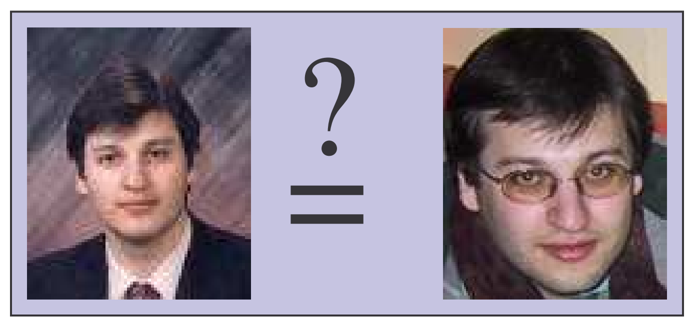
- Secure cryptographic keys: long, random strings
- Too hard to remember
- Keep copy under doormat?
- Use short, easy-to-remember password (PIN)?
  - easy to remember = easy to guess

Passwords You Won’t Forget

- Personal Info
  - Mom’s maiden name, Date of birth
  - Name of first pet, name of street where you grew up
- Biometrics
  - Fingerprints
  - Iris Scan
  - Face recognition
  - Hand Geometry
  - Voice print
  - Signature







Issues with Personal Info / Biometrics

➢ Noise and human error

- Fingerprint is Variable
  - Finger orientation, cuts, scrapes
- Personal info subject to memory failures / format
  - Name of your first girl/boy friend:
    "Um… Catherine? Katharine? Kate? Sue?"
- Measured data will be "close" to original
  in some **metric** (application-dependent)

Issues with Personal Info / Biometrics

- Noise and human error
- Not Uniformly Random

- (Crypto keys should be random)
- Fingerprints are represented as list of features
  - All fingers look similar
- Distribution is unknown
  - Ivanov is rare last name… unless first name is Sergei

Issues with Personal Info / Biometrics

- Noise and human error
- Not Uniformly Random
- Should Not Be Stored in the Clear
  - Theft is easy, makes info useless
    - Customer service representatives learn Mom's maiden name, Social Security Number, …
  - Keys cannot be changed many times
    - 10 fingers, 2 eyes, 1 mother

Issues with Personal Info / Biometrics

- Noise and human
- Not
- Sh

Do we ***want*** to use this stuff as a password?

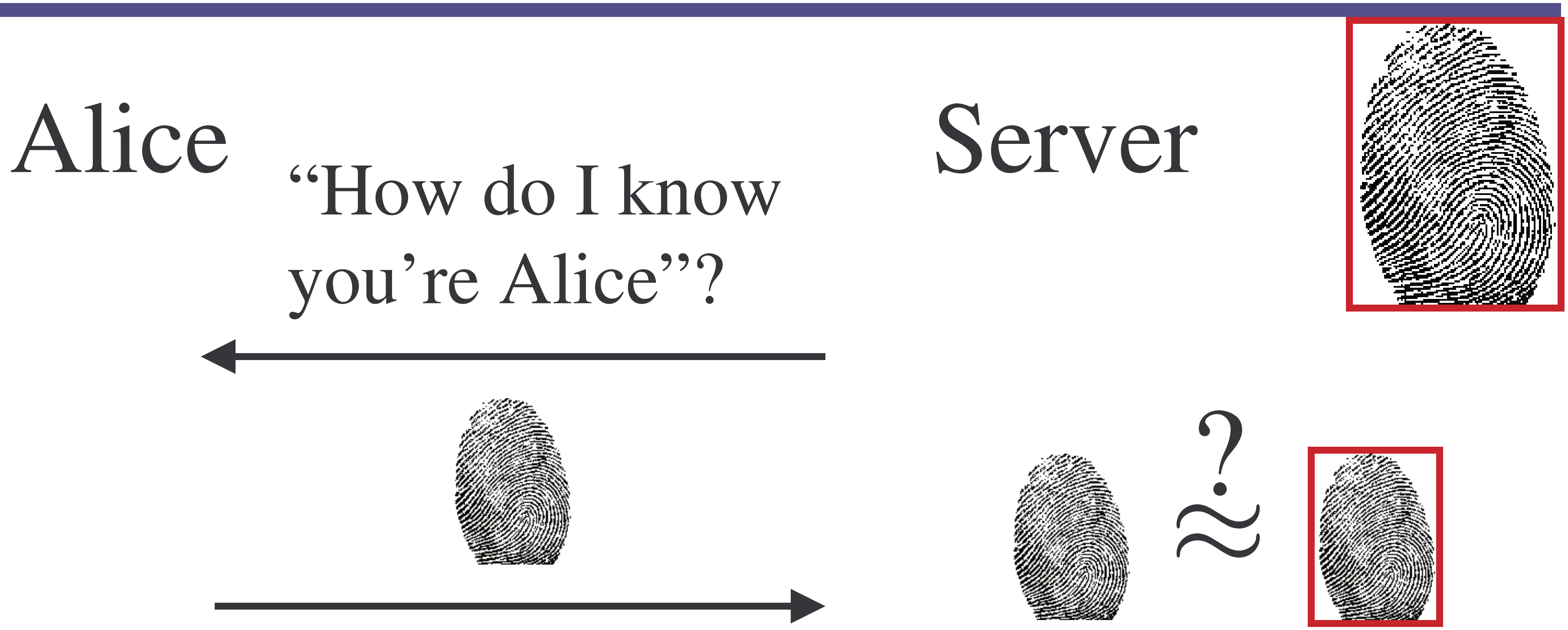
Issues with Personal Info / Biometrics

- Noise and human error
- Not Uniformly Random
- Should Not Be Stored in the Clear

How can we use this stuff as a password?

Example: Authentication

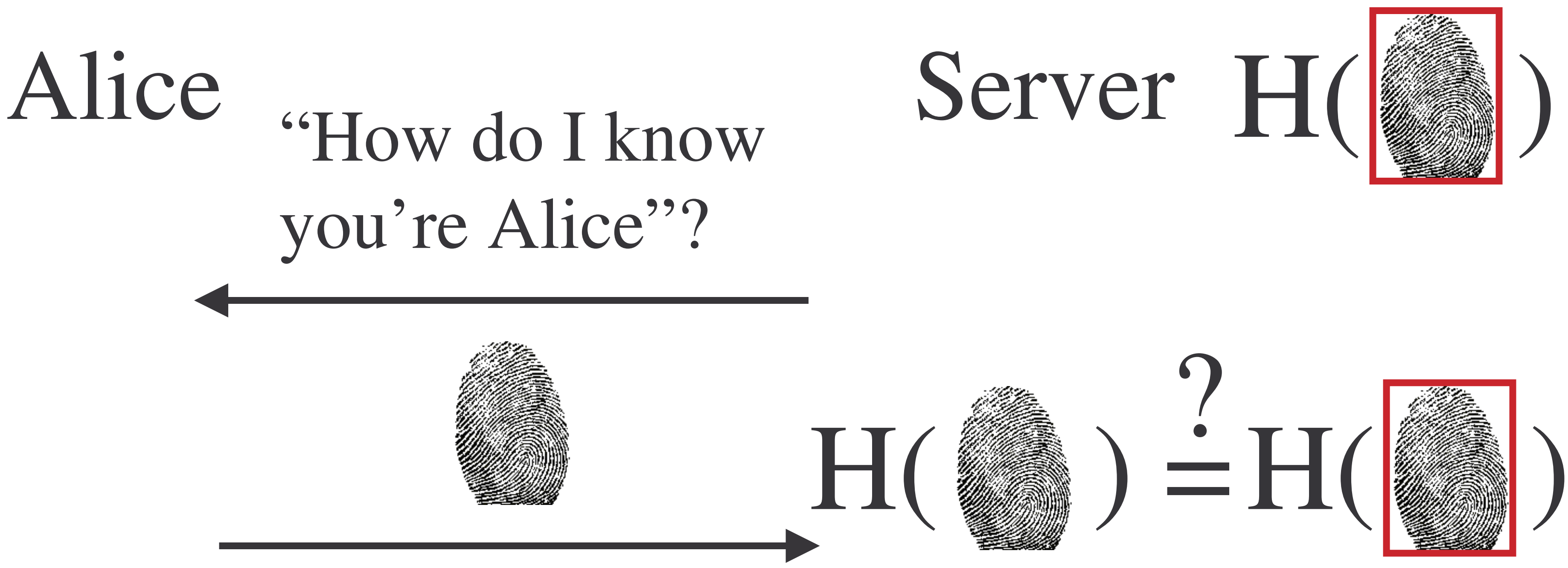
Authentication [EHMS,JW]



Solution #1: Store a copy on server

Problem: Password in the Clear

Authentication [EHMS,JW]



Solution #2: Store a hash of password

Problem: No Error Tolerance

This Talk

Formal framework and new constructions

for handling noisy key material

Provable Security

Related Work

Basic set-up studied for quite a while, lots of nice ideas:

- Davida, Frankel, Matt ’98, Ellison, Hall, Milbert, Shneier ’00

First abstractions:

- Juels, Wattenberg ‘99, Frykholm, Juels ’01
  - Handling noisy data in Hamming metric
- Juels, Sudan ’02
  - Set difference metric

Provable security:

- Linnartz, Tuyls ’03
  - Provable security, specific distribution (multivariate Gaussian)

This Talk

Formal framework and new constructions

for handling noisy key material

Provable Security

Outline

- ☑ Basic Setting: Password Authentication
- ❑ Simple abstraction: Secure Sketch
  - Example: Hamming distance
  - Secure Sketch $\Rightarrow$ Authentication
- ❑ Constructions for "set difference" distance
- ❑ Other schemes via metric embeddings: edit distance
- ❑ Privacy for Stored Data

Outline

- ❑ Basic Setting: Password Authentication
- ❑ Simple abstraction: Secure Sketch
  - Example: Hamming distance
  - Secure Sketch $\Rightarrow$ Authentication
- ❑ Constructions for "set difference" distance
- ❑ Other schemes via metric embeddings: edit distance
- ❑ Privacy for Stored Data

Secure Sketch

$$x \longrightarrow \boxed{S} \longrightarrow S(x)$$

1. **Error-correction**: If x' is "close" to x, then recover x

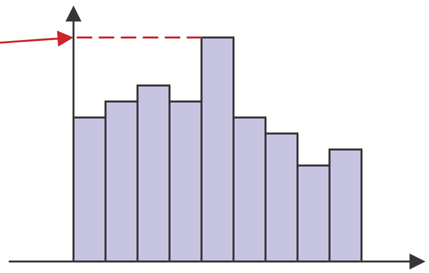
$$\begin{array}{l} x' \longrightarrow \\ S(x) \longrightarrow \end{array} \boxed{\text{Recover}} \longrightarrow x$$

2. **Secrecy**: Given $S(X)$, it's hard to predict X

- Meaning of "close" depends on application
- Secrecy: loss of min-entropy

Measuring Security

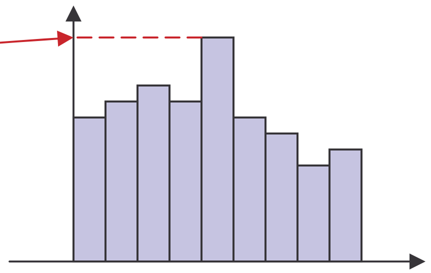
- X a random variable on $\{0,1\}^n$
- Probability of predicting X = $\max_x \Pr[X = x]$



- There are various ways to measure entropy…
- **Min-entropy**: $H_\infty(X) = -\log\left(\max_x \Pr[X=x]\right)$
- Uniform on $\{0,1\}^n$: $H_\infty(U_n) = n$
- " Password has min-entropy t " means that adversary's probability of guessing the password is 2^{-t}
- Passwords had better have high entropy!

Measuring Security

- X a random variable on $\{0,1\}^n$
- Probability of predicting $X = \max_x \Pr[X = x]$



- There are various ways to measure entropy…
- **Min-entropy**: $H_\infty(X) = -\log\left(\max_x \Pr[X=x]\right)$
- Conditional entropy

$$H_\infty(X \mid Y) = -\log(\text{ prob. of predicting } X \text{ given } Y\,)$$

$$= -\log\left(\, \mathrm{Exp}_y \left\{ \max_x \Pr[X=x \mid Y=y] \right\} \right)$$

Secure Sketch

1. **Error-correction**: If x' is "close" to x, then recover x

2. **Secrecy**: Given $S(X)$, it's hard to predict X

Goals: - Minimize entropy loss: $H_\infty(X) - H_\infty(X \mid S(X))$

- Maximize tolerance: how "far" x' can be from x

Example: Code-Offset Construction

[BBR88, Cré97,…, JW02]

Code-Offset Construction [BBR,Cré,JW]

- View password as n bit string : $\boldsymbol{x} \in \{0,1\}^n$
- Error model: small number of flipped bits
- Hamming distance:

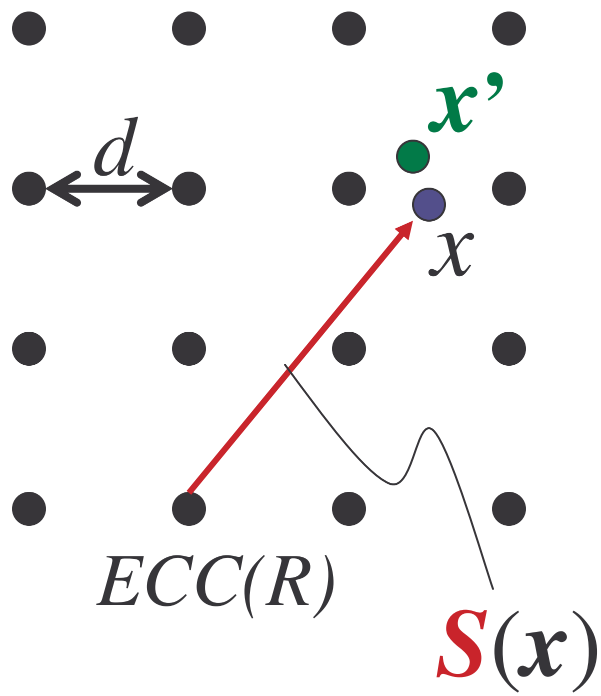
  $d_H(\boldsymbol{x},\boldsymbol{x'})$ = # of positions in which $\boldsymbol{x}$, $\boldsymbol{x'}$ differ
- Main idea: non-conventional use of standard error-correcting codes

Code-Offset Construction [BBR,Cré,JW]

- Error-correcting code ECC: k bits $\rightarrow$ n bits
- Any two codewords differ by at least d bits
- $S(x) = x \oplus \mathrm{ECC}(R)$

  where R is random string

  Equiv: $S(x)$ = syndrome(x)

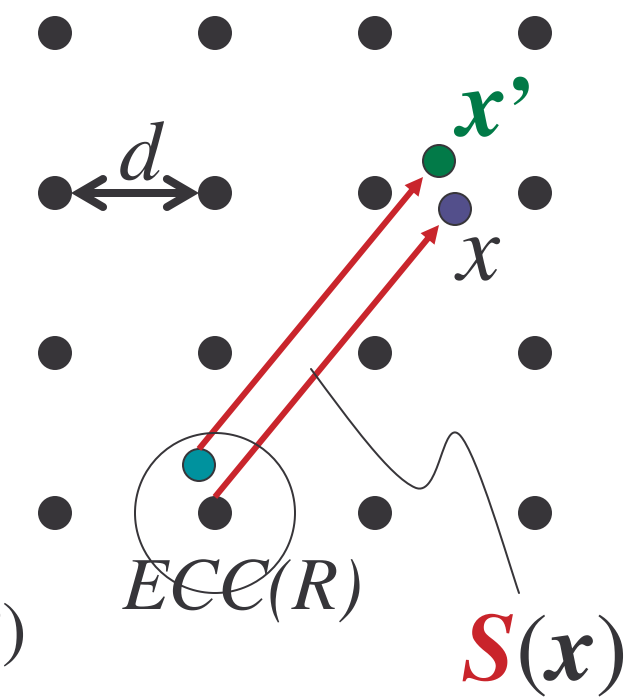


- Corrects $d/2$ errors
- How much entropy loss?

- Error-correcting code ECC: k bits $\rightarrow$ n bits
- Any two codewords differ by at least d bits
- $S(\boldsymbol{x}) = \boldsymbol{x} \oplus \mathrm{ECC}(R)$

  where R is random string

  Equiv: $S(\boldsymbol{x}) = \mathrm{syndrome}(\boldsymbol{x})$
- Given $S(\boldsymbol{x})$ and $\boldsymbol{x}'$ close to $\boldsymbol{x}$:
  - Compute $\boxed{\boldsymbol{x}' \oplus S(\boldsymbol{x})}$
  - Decode to get ECC(R)
  - Compute $\boldsymbol{x} = S(\boldsymbol{x}) \oplus \mathrm{ECC}(R)$



Revealing n bits costs $\leq n$ bits of entropy

- Error-correcting code ECC: k bits $\rightarrow$ n bits
- Any two codewords differ by at least d bits
- $\boldsymbol{S}(\boldsymbol{x}) = \boldsymbol{x} \oplus \mathrm{ECC}(R)$

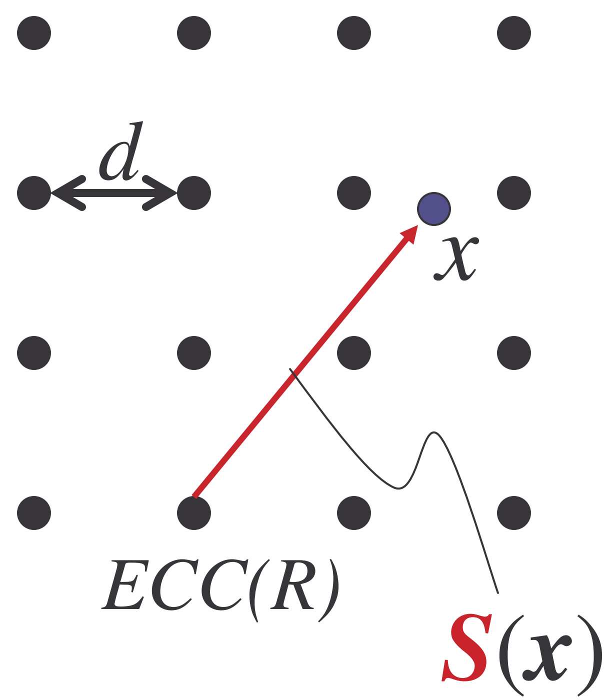
  where R is random string

$$H_\infty(X \mid \boldsymbol{S}(X))$$

$$= H_\infty(X, R \mid \boldsymbol{S}(X))$$

$$\geq H_\infty(X) + H_\infty(R) - |\boldsymbol{S}(X)|$$

$$= H_\infty(X) + k - n$$

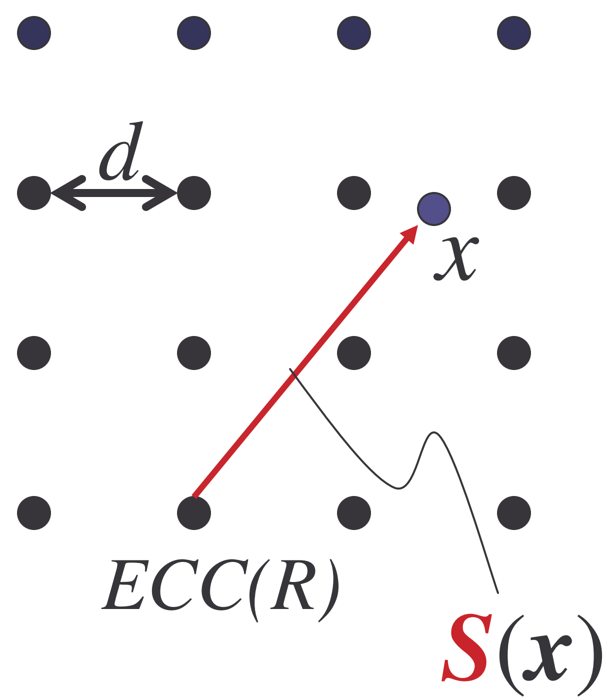


$$\text{Entropy loss } = n - k$$
$$= \text{redundancy of code}$$

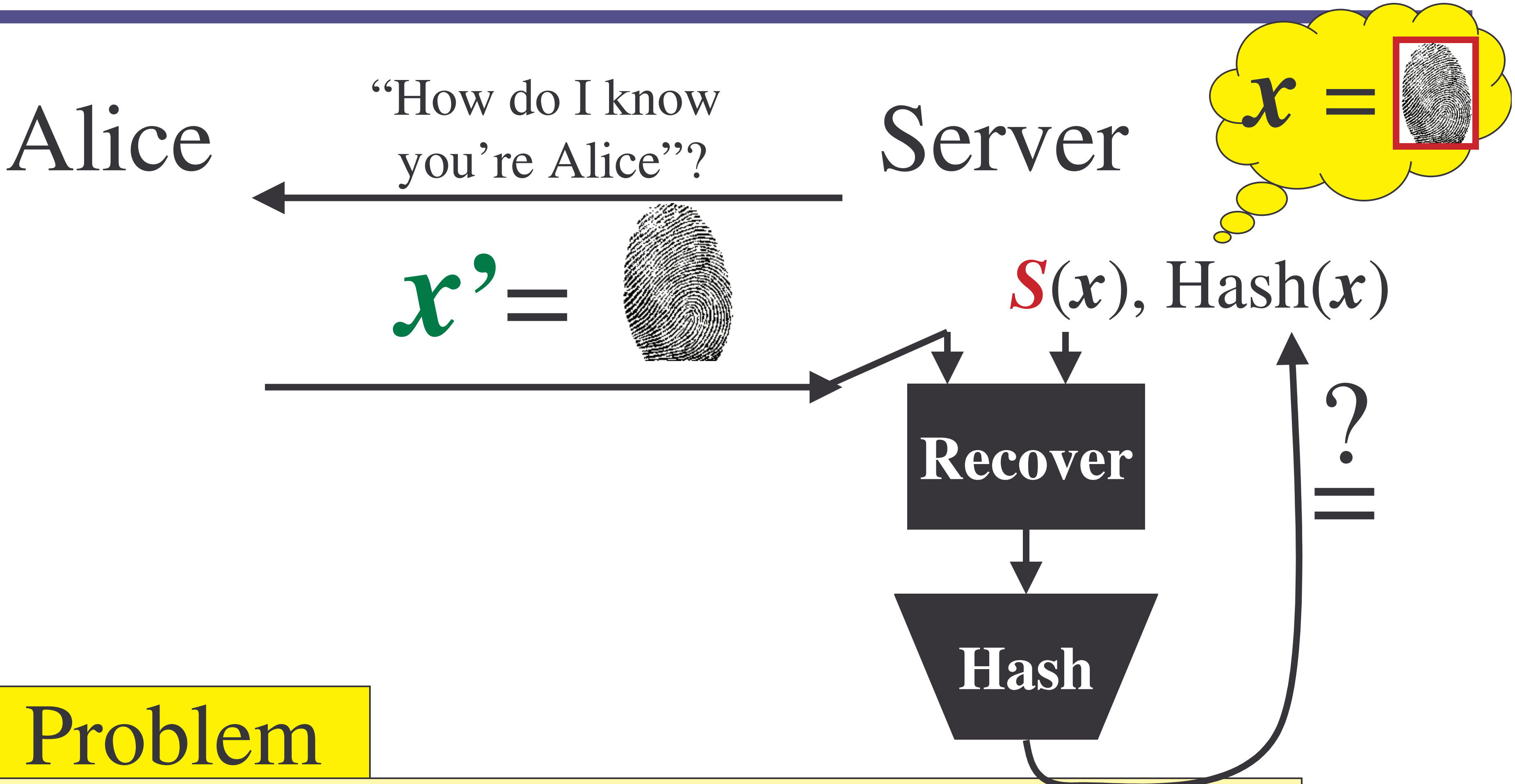
- Error-correcting code ECC: k bits $\to$ n bits
- Any two codewords differ by at least d bits
- $\boldsymbol{S}(\boldsymbol{x}) = \boldsymbol{x} \oplus \mathrm{ECC}(R)$

  where R is random string

$$H_\infty(X \mid \boldsymbol{S}(X))$$
$$= H_\infty(X, R \mid \boldsymbol{S}(X))$$
$$\geq H_\infty(X) + H_\infty(R) - |\boldsymbol{S}(X)|$$
$$= H_\infty(X) + k - n$$



Using Sketches for Authentication



Problem

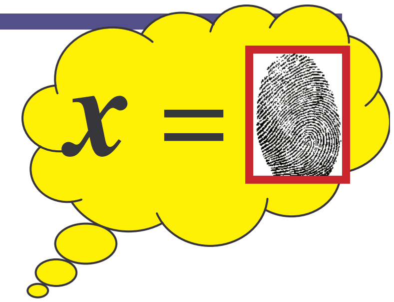
- Input to Hash should be uniformly random
- X is not uniform (especially given $S(X)$)

Using Sketches for Authentication

Alice

"How do I know you're Alice"?

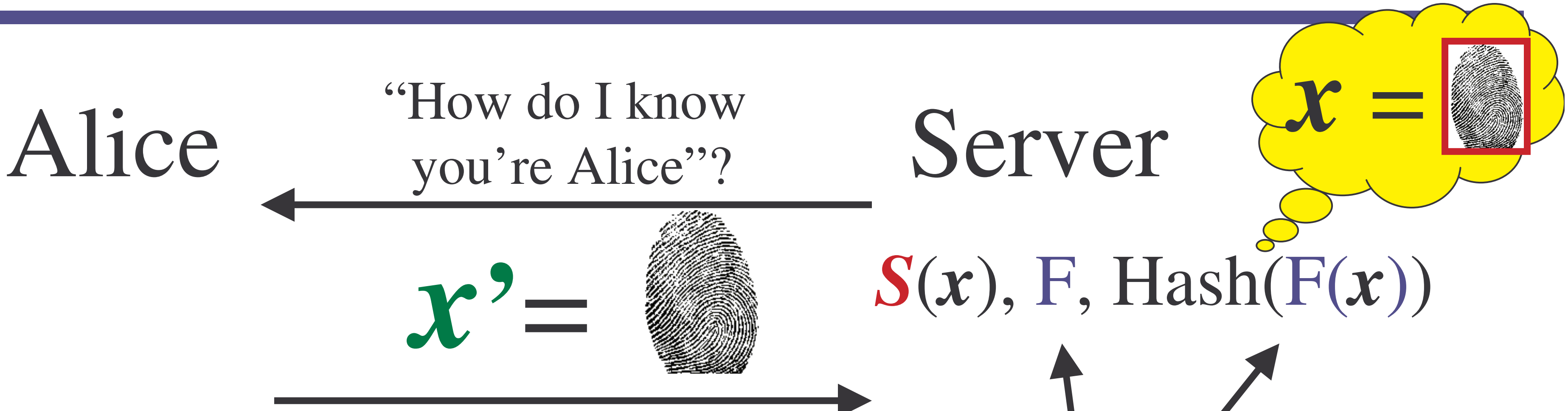
Server



x' = 

$S(x)$, Hash(x)

Padding via Random Functions



Solution

Scramble input with "random" function ***F***. Store ***F***.

- 2-universal hash is sufficient (e.g. random linear map)
- (Any "strong extractor" also works)
- When is this secure?

Padding via Random Functions

Alice

"How do I know you're Alice"?

Server

$x =$ 

$x' =$ 

$S(x)$, F, Hash(F(x))

Solution

Scramble input with "random" function F. Store F.

- Secure as long as:

$$\text{Entropy-Loss}_S + |\text{Hash}| + 2\log(1/\varepsilon) \leq H_\infty(X)$$

- Proof idea: $S(x)$, F, Hash(F(x)) $\approx$ $S(x)$, F, Hash(R)
- Similar to "left-over hash lemma / privacy amplification"

Sketches and Authentication

- "Secure Sketch" + Hashing Solves Authentication
- "Hamming" errors can be handled with standard ECC
- **Assumption**: X has high entropy
  - Necessary
  - X could be several passwords taken together
- Similar techniques imply one can use X as key for many crypto applications (e.g. encryption)
  - Covers several previously studied settings

Outline

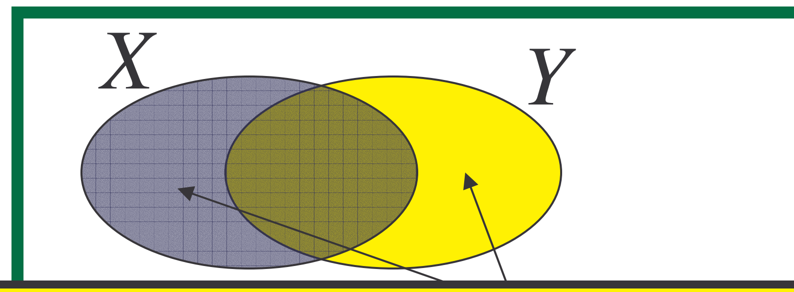
- ❑ Basic Setting: Password Authentication
- ❑ Simple abstraction: Secure Sketch
  - Example: Hamming distance
  - Secure Sketch ⇒ Authentication
- ❑ "Set difference" distance
- ❑ Other schemes via metric embeddings: edit distance
- ❑ Privacy for Stored Data

Why Set Difference? [EHMS,FJ,JS]

- Inputs: tiny subsets in a HUGE universe
- Some representations of personal / biometric data:
  - Fingerprints represented as feature list (minutiae/ridge meetings)
  - List of favorite books
- $\boldsymbol{X} \subseteq \{1,\ldots,N\}$, $\#\boldsymbol{X} = s$
- $d_S(\boldsymbol{X},\boldsymbol{Y}) = ½\,\#\,(\boldsymbol{X}\ \Delta\ \boldsymbol{Y})$

  = Hamming distance

  on vectors in $\{0,1\}^N$.



- Code-offset not good: N-bit string is too long!
- Want: $s \log N$ bits

Recall: Secure Sketch

$$X \longrightarrow \boxed{\boldsymbol{S}} \longrightarrow \boldsymbol{S}(\mathrm{X})$$

1. **Error-correction**: If $\boldsymbol{x}$' is "close" to $\boldsymbol{x}$, then

$$\begin{matrix} \boldsymbol{x}' \longrightarrow \\ \boldsymbol{S}(\boldsymbol{x}) \longrightarrow \end{matrix} \boxed{\text{Recover}} \longrightarrow \boldsymbol{x}$$

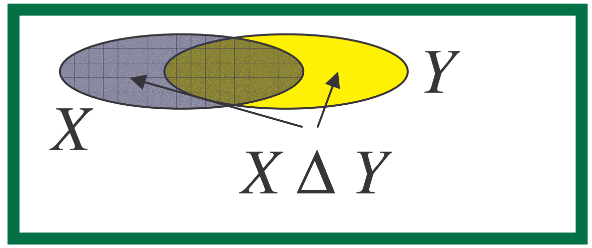
2. **Secrecy**: Given $\boldsymbol{S}(X)$, it's hard to predict X

Goals: - Minimize entropy loss: $H_\infty(X) - H_\infty(X \mid \boldsymbol{S}(X))$

- Maximize tolerance: how "far" $\boldsymbol{x}$' can be from $\boldsymbol{x}$

New Constructions for Set Difference

- $X \subseteq \{1,\ldots,N\}$, $\#X = s$, $d_S(X,Y) = ½ \# (X \Delta Y)$
- Two constructions



1. punctured Reed-Solomon code
2. Sublinear-time decoding of BCH codes from syndromes

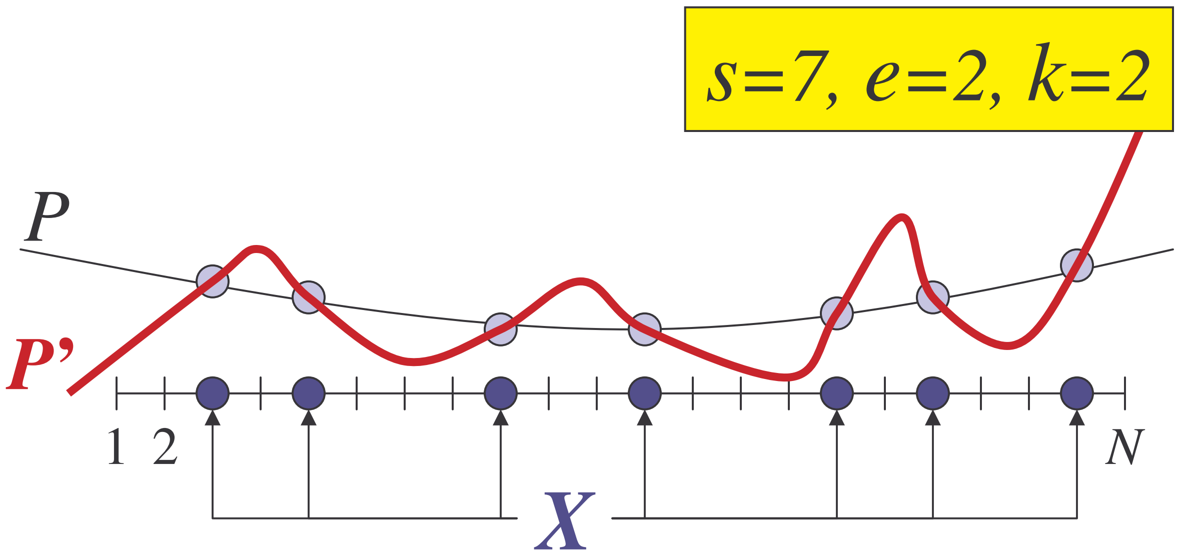
- Both constructions:
  - As good as code-offset could be (≈ optimal)
  - Storage space $\leq (s + 1) \log N$
  - Entropy loss $\mathbf{2\,e \log N}$ to correct e errors
  - Improve previous best [JS02] (+ analysis)

Reed-Solomon-based Sketch

$X \subseteq \{1,\ldots,N\}$, $\#X = s$, $d_S(X,Y) = ½ \# (X \Delta Y)$

Suppose N is prime, work in $\mathbf{Z}_N$

1. $k := s - 2e - 1$
2. Pick random poly. $P()$ of degree $\leq k$
3. ***P'***() **:=** monic degree s poly. s.t. ***P'***(z)=$P(z)$ $\forall z \in X$
4. Output ***S***(*X*)**=*P'***

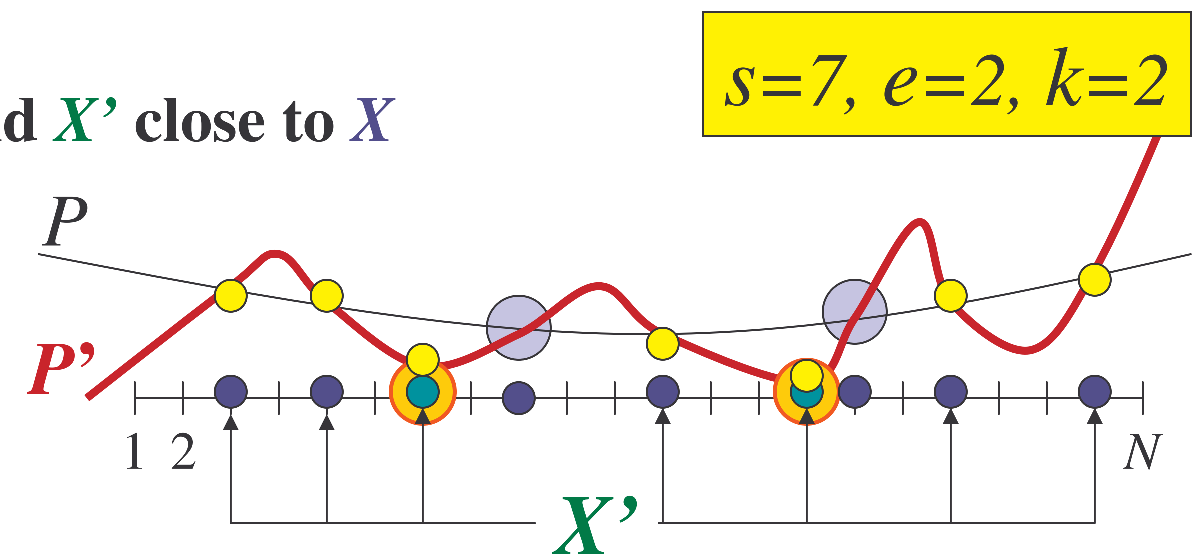


Reed-Solomon-based Sketch

$X \subseteq \{1,\ldots,N\}$, $\#X = s$, $d_S(X,Y) = ½\,\#\,(X \,\Delta\, Y)$

Suppose N is prime, work in $\mathbf{Z}_N$

1. $k := s - 2\,e - 1$
2. Pick random poly. $P()$ of degree $\leq k$
3. $P'()$:= monic degree s poly. s.t. $P'(z)=P(z)\ \forall z \in X$
4. Output $S(X)=P'$



Recovery: Given P' and X' close to X

1. Reed-Solomon decoding yields P
2. Intersections of P and P' yield X

Reed-Solomon-based Sketch

$X \subseteq \{1,\ldots,N\}$, $\#X = s$, $\quad d_S(X,Y) = ½\,\#\,(X \,\Delta\, Y)$

Suppose N is prime, work in $\mathbf{Z}_N$

1. $k := s - 2e - 1$
2. Pick random poly. $P()$ of degree $\leq k$
3. $P'()$:= monic degree s poly. s.t. $P'(z)=P(z)\ \forall z \in X$
4. Output $S(X)=P'$

Entropy loss:

$$H_\infty(X \mid P') = H_\infty(X,P \mid P') \geq H_\infty(X) + H_\infty(P) - |P'|$$

$$= H_\infty(X) + (k+1)\log N - s \log N$$

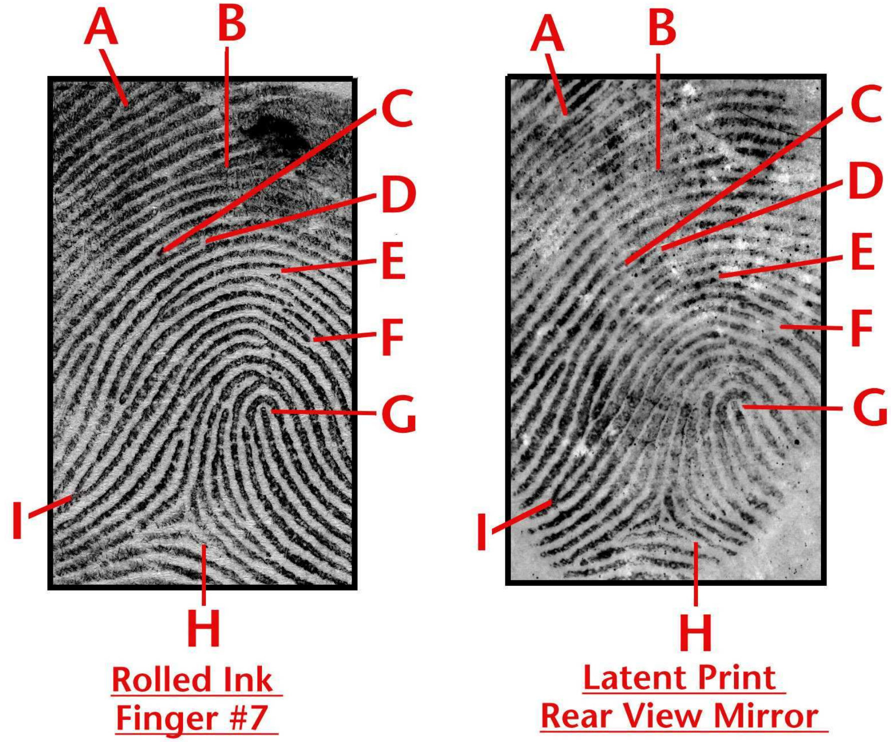
$$= H_\infty(X) - 2e \log N$$

Outline

- ☑ Basic Setting: Password Authentication
- ☑ Simple abstraction: Secure Sketch
  - Example: Hamming distance
  - Secure Sketch $\Rightarrow$ Authentication
- ☑ "Set difference" distance: Reed-Solomon construction
- ❑ Other schemes via metric embeddings: edit distance
- ❑ Privacy for Stored Data

Other metrics?

- Real error models not as clean as Hamming & set diff.
- Algebraic techniques won't apply directly.



Other metrics?

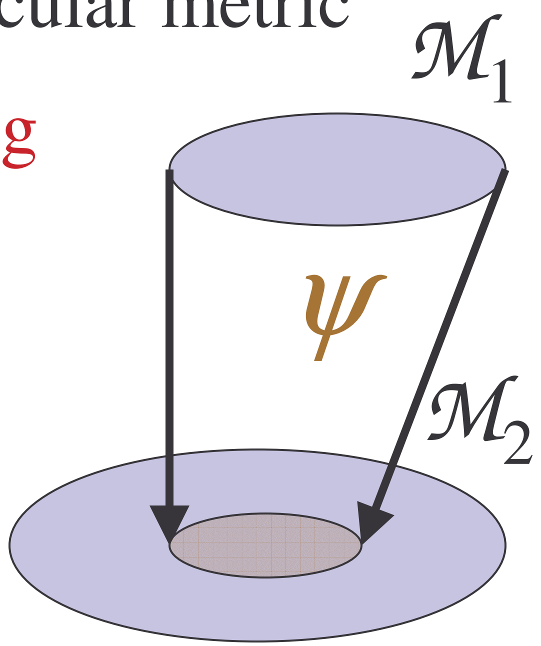
- Real error models not as clean as Hamming & set diff.
- Algebraic techniques won't apply directly.
- Possible Approaches:

1. Develop new scheme tailored to particular metric
2. Reduce to easier metric via embedding

$$\psi: \mathcal{M}_1 \to \mathcal{M}_2$$

$$\boldsymbol{x}, \boldsymbol{y} \text{ close} \Rightarrow \psi(\boldsymbol{x}), \psi(\boldsymbol{y}) \text{ close}$$

$$\boldsymbol{x}, \boldsymbol{y} \text{ far} \Rightarrow \psi(\boldsymbol{x}), \psi(\boldsymbol{y}) \text{ far}$$



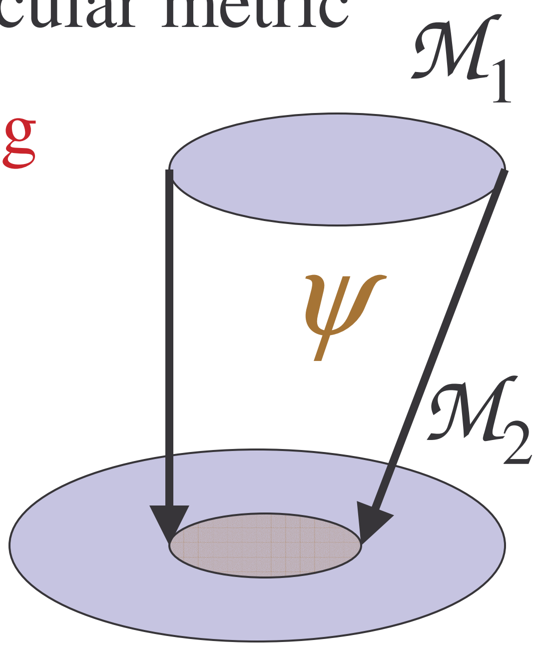
*Bio*metric embeddings

- Real error models not as clean as Hamming & set diff.
- Algebraic techniques won't apply directly.
- Possible Approaches:

1. Develop new scheme tailored to particular metric
2. Reduce to easier metric via embedding

$$\psi : \mathcal{M}_1 \to \mathcal{M}_2$$

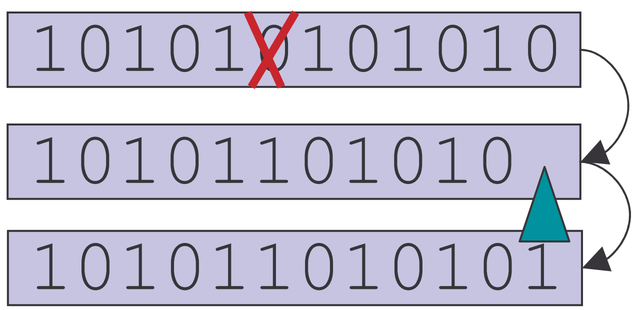
$$\boldsymbol{x}, \boldsymbol{y} \text{ close} \Rightarrow \psi(\boldsymbol{x}), \psi(\boldsymbol{y}) \text{ close}$$



A is a large set $\Rightarrow \psi(A)$ is large
$H_\infty(A)$ large $\Rightarrow H_\infty(\psi(A))$ large

Edit Distance (suggested by P. Indyk)

- Strings of bits
- d(x,y) = number of insertions & deletions to go from x to y



- Good standard embeddings into Hamming not known
- Shingling [Broder]: "biometric" embedding into Set.Diff.

Outline

- ☑ Basic Setting: Password Authentication
- ☑ Simple abstraction: Secure Sketch
  - Example: Hamming distance
  - Secure Sketch ⇒ Authentication
- ☑ "Set difference" distance: Reed-Solomon construction
- ☑ Other schemes via metric embeddings: edit distance
- ❑ Privacy for Stored Data

[DS03]

Stronger Privacy?

- Previous notion: Unpredictability
  - Can't guess X even after seeing sketch
  - Sufficient for using X as a crypto key
- What about the privacy of X itself?
  - Do not want particular info about X leaked (say, first 20 bits)
- Ideal notion:

$$X \text{ almost independent of } S(X)$$

- **Problem**: Some info must be leaked by $S(X)$ (provably)
  - Mutual information $I(X\,;\,S(X))$ is large
- We want the ensure that "useful" information is hidden

Hiding All Functions ([CMR], à la [GM])

Definition: $S(X)$ hides all functions of X if

For all functions g, for all adversaries A, $\exists\, A'$

$$\Pr[\, A(S_1(X), S_2(X), \ldots) = g(X)\,] - \Pr[\, A'() = g(X)\,] < \varepsilon$$

Intuition: *"A cannot guess g(X) given polynomially-many copies of* $S(X)$*"*

$$\begin{tikzcd} X \arrow[r] \arrow[d] & S_1(X), S_2(X), \ldots \arrow[dl, dashed, "\text{difficult}"] \\ g(X) & \end{tikzcd}$$

Implies unpredictability

Hiding All Functions ([CMR], à la [GM])

Definition: $S(X)$ hides all functions of X if

For all functions g , for all adversaries A, $\exists\, A'$

$$\Pr[\, A(S_1(X), S_2(X), \ldots) = g(X)\,] - \Pr[\, A'() = g(X)\,] < \varepsilon$$

- No known constructions satisfy this
  - (Some recent ideas by [vDW])
- Our results:
  - Information-theoretically secure (vs computational)
  - One use only

One-time Security [CMR,RW]

Definition: $S(X)$ hides all functions of X if

For all functions g, for all adversaries A, $\exists A'$

$$\Pr[\ A(S(X)) = g(X)\ ] - \Pr[\ A'() = g(X)\ ] < \varepsilon$$

One copy of $S(X)$

$$\begin{tikzcd} X \arrow[r] \arrow[d] & S(X) \arrow[dl, dashed, "\text{difficult}"] \\ g(X) & \end{tikzcd}$$

One-time Security [CMR,RW]

Definition: $\boldsymbol{S}(\boldsymbol{X})$ hides all functions of $\boldsymbol{X}$ if

For all functions g , for all adversaries A, $\exists A'$

$$\Pr[\,A(\boldsymbol{S}(\boldsymbol{X})) = g(\boldsymbol{X})\,] - \Pr[\,A'() = g(\boldsymbol{X})\,] < \varepsilon$$

- Can be achieved in code-offset construction
  - Use randomly chosen code from some family

$$\boldsymbol{S}(\boldsymbol{x}) = \text{description of ECC}, \boldsymbol{x} \oplus \text{ECC}(R)$$

  - Need to keep decodability
  - Exact parameters still unknown

Technique: Equivalence to Extraction

Definition: $S(X)$ hides all functions of X if

For all functions g , for all adversaries A, $\exists A'$

$$\Pr[\, A(S(X)) = g(X)\,] - \Pr[\, A'() = g(X)\,] < \varepsilon$$

$S()$ is an "extractor" if for all r.v.'s X_1, X_2 of min-entropy t

$$S(X_1) \approx_\varepsilon S(X_2)$$

Thm: $S(X)$ hides all functions of X, whenever $H_\infty(X) \geq t$

$\Leftrightarrow$ $S()$ is an "extractor" for r.v.'s of min-entropy $t-1$

Outline

- ☑ Basic Setting: Password Authentication
- ☑ Simple abstraction: Secure Sketch
  - Example: Hamming distance
  - Secure Sketch $\Rightarrow$ Authentication
- ☑ "Set difference" distance: Reed-Solomon construction
- ☑ Other schemes via metric embeddings: edit distance
- ☑ Privacy for Stored Data

Conclusions

- Generic framework for turning noisy, non-uniform data into secure cryptographic keys
  - Abstraction, simplicity allow comparing schemes
  - Constructions for Hamming, Set Difference, Edit Metrics
  - Progress towards strong privacy of data

Conclusions

- Generic framework for turning noisy, non-uniform data into secure cryptographic keys
- New techniques for information-theoretic crypto
  - Non-standard use of extractors
  - Connections to coding theory, embeddings

Conclusions

- Generic framework for turning noisy, non-uniform data into secure cryptographic keys
- New techniques for information-theoretic crypto
- Applications to other settings
  - perfect one-way functions,
  - encryption of high-entropy messages [DS03],
  - bounded-storage crypto [DV04],
  - physically uncloneable functions [DLD04]

Conclusions

- Generic framework for turning noisy, non-uniform data into secure cryptographic keys
- New techniques for information-theoretic crypto
- Applications to other settings
- Future work
  - Other “metrics”
  - Stronger privacy (computational version a la [CMR98])
  - Reusability

Conclusions

- Generic framework for turning noisy, non-uniform data into secure cryptographic keys
- New techniques for information-theoretic crypto
- Applications to other settings
- Future work
- Bigger Picture?

Biometric "Security" Wide Open

- This talk: Storage
- Other vulnerabilities:
  - Spoofing
  - Hardware must be secure
- Bigger threat to privacy comes from misuse/overuse
  - Function creep (SSN)
  - Not revocable
  - Can they be kept secret even in principle?

Questions?

edit distance

- dis(x,y) = number of insertions & deletions to go from x to y
- E.g., typos in a passphrase

x = `Albuquerque-Massachusetts-Winnipesaukee`
y = `Albuqurque-Masachusetts-Winipesssaukee`

$$\text{dis}(x,y) = 4$$

- Idea: convert to set difference via shingling [Broder]
- Map a string to a set of all its length-c substrings

shingling for fuzzy extractors

- View string as set of shingles
- c-shingling
  - each edit error gives c set errors
  - entropy loss $(n/c) \log n$, where n is input string length
- Optimize c
- If $H_\infty(W) = \Theta(n)$, can extract $\Theta(n)$ bits tolerating $\Theta(n / \log^2 n)$ errors

x	y
`Albuq`	`Albuq`
`lbuqu`	`lbuqu`
`buque`	`buqur`
`uquer`	`uqurq`
`querq`	`qurqu`
`uerqu`	`urque`
`erque`	
`rque-`	`rque-`
`que-M`	`que-M`
`ue-Ma`	`ue-Ma`
`e-Mas`	`e-Mas`
`-Mass`	`-Masa`
`Massa`	`Masac`

$x =$ `Albuquerque-Massachusetts-Winnipesaukee`

$y =$ `Albuqurque-Masachusetts-Winipessaukee`

Other Slides

The Problem: We’re Human

“Humans are incapable of securely storing high-quality cryptographic keys, and they have unacceptable speed when performing cryptographic operations. (They are also large, expensive to maintain, difficult to manage, and they pollute the environment. […] But they are sufficiently pervasive that we must design our protocols around their limitations.)”

From *Network Security* by Kaufman, Perlman and Speciner.

Stuff I want to say

- generic framework
- general tools
- About authentication: interplay between computational assumptions and information-theoretic technique
- practical… may be implemented
- General context: provable security

*Bio*metric embeddings

$$\psi: \mathcal{M}_1 \to \mathcal{M}_2$$

We care about entropy: non-standard requirements

- $\boldsymbol{x,y}$ close $\Rightarrow \psi(\boldsymbol{x}),\ \psi(\mathbf{y})$ close

$$\frac{d_1(\boldsymbol{x,y})}{d_2(\psi(\boldsymbol{x}),\ \psi(\mathbf{y}))} \leq \alpha$$

- A is a large set $\Rightarrow \psi(A)$ is large

$$\frac{\#\,\boldsymbol{A}}{\#\,\psi(\boldsymbol{A})} \geq \beta \quad \text{or, equivalently} \quad \frac{H_\infty(\boldsymbol{X})}{H_\infty(\psi(\boldsymbol{X}))} \geq \beta$$

Statistical Distinguishability

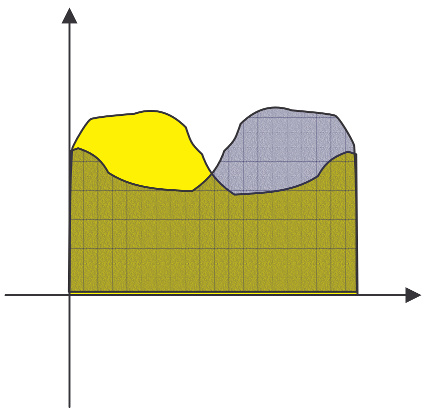
- Statistical Difference (L_1): For distributions $p_0(x)$, $p_1(x)$:

$$SD(p_0,p_1) = \tfrac{1}{2} \sum_x | p_0(x) - p_1(x) |$$

- SD measures distinguishability:

  If $b \leftarrow \{0,1\}$, $x \leftarrow p_b$ then

  $\max_A | \Pr[A(x)=b] - \tfrac{1}{2} | = \tfrac{1}{2} SD(p_0,p_1)$



- (Notation: $A \approx_\varepsilon B$ if $SD(A,B) \leq \varepsilon$)

Statistical Distinguishability

- Two probability distributions $p_0(x)$, $p_1(x)$

Sphinx:

1. Flips a fair coin
2. - **Heads**: Samples Z according to p_0
   - **Tails**: Samples Z according to p_1
3. Shows Z to Greek Hero

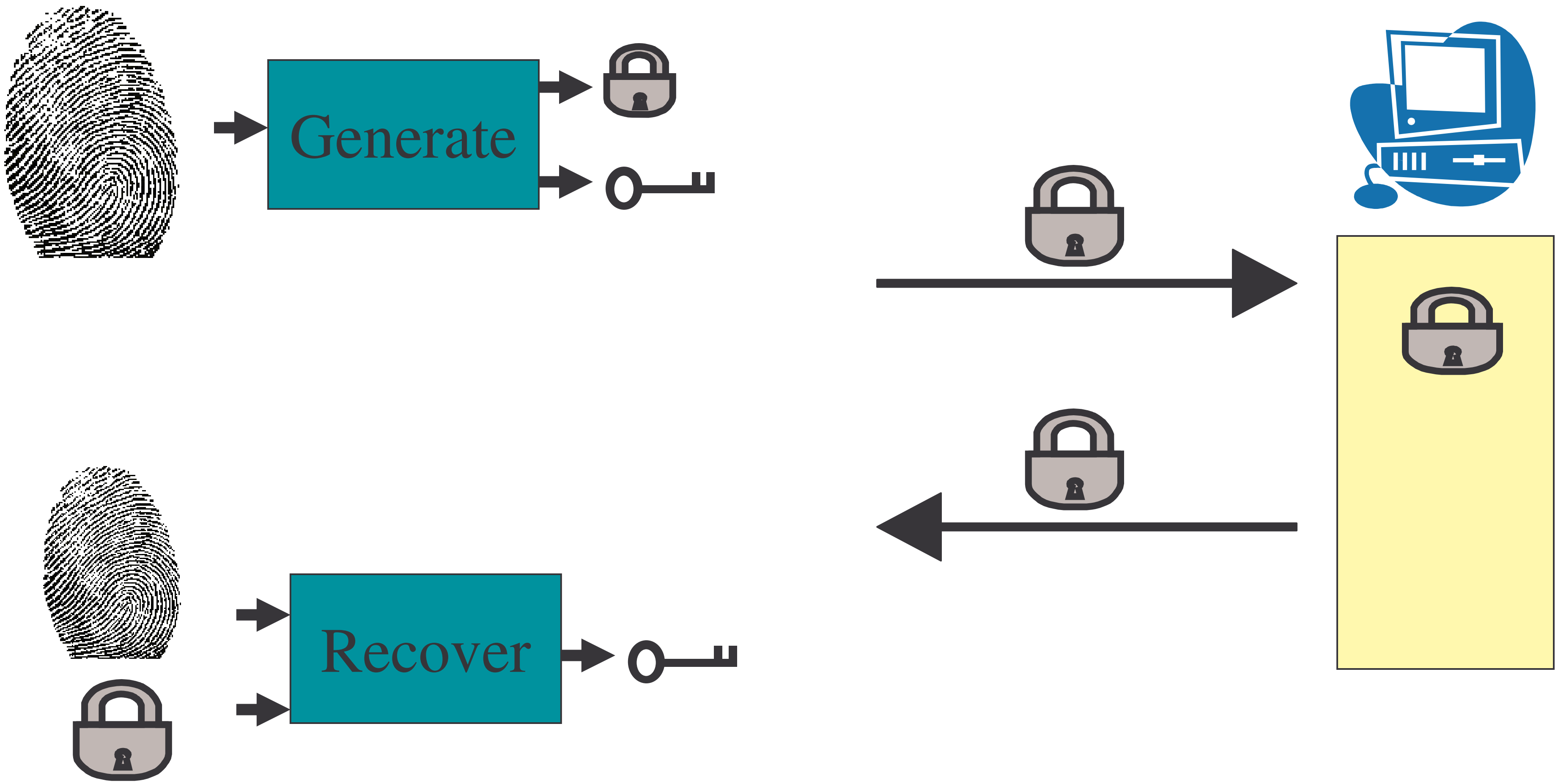
Greek Hero: Guesses if coin was heads or tails.

Hero can wins with probability at least ½

Hero wins w. prob. **½ + ε** $\Rightarrow$ p_0, p_1 are **ε**-distinguishable

Key Recovery [EHMS, FJ, JS]

www.Fingers2Keys.com



Lemma

If

- $\boldsymbol{F}$:$\{0,1\}^n \to \{0,1\}^N$ chosen from 2-wise indep. hash f'ly

  (N can be arbitrarily large)

- h: $\{0,1\}^N \to \{0,1\}^k$ any function
- $\boldsymbol{X}$, $\boldsymbol{Y}$ such that $\boldsymbol{X} \in \{0,1\}^n$ and $H_\infty(\boldsymbol{X}|\boldsymbol{Y}) \geq k + 2\log(1/\varepsilon)$

Then

$$\boldsymbol{Y}, \boldsymbol{F}, h(\boldsymbol{F}(\boldsymbol{X})) \approx_\varepsilon \boldsymbol{Y}, \boldsymbol{F}, h(\boldsymbol{R})$$