Hash Functions

Hash Functions

Lecture 8

Hash Functions

Lecture 8

Before we talk about digital signatures...

A Tale of Two Boxes

A Tale of Two Boxes

- Much of today's applied cryptography works with two magic boxes

A Tale of Two Boxes

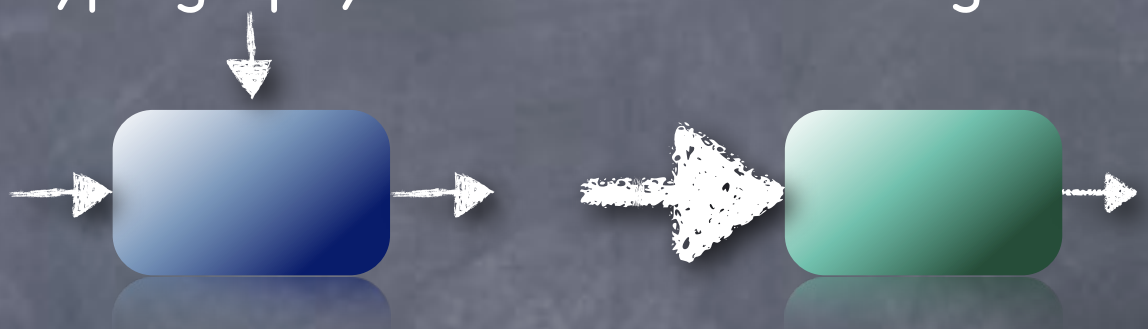
- Much of today's applied cryptography works with two magic boxes
  - Block Ciphers

A Tale of Two Boxes

- Much of today's applied cryptography works with two magic boxes
  - Block Ciphers

A Tale of Two Boxes

- Much of today's applied cryptography works with two magic boxes
  - Block Ciphers
  - Hash Functions



A Tale of Two Boxes

- Much of today's applied cryptography works with two magic boxes
  - Block Ciphers
  - Hash Functions

- Block Ciphers: Best modeled as (strong) Pseudorandom Permutations, with inversion trapdoors

A Tale of Two Boxes

- Much of today's applied cryptography works with two magic boxes
  - Block Ciphers
  - Hash Functions

- Block Ciphers: Best modeled as (strong) Pseudorandom Permutations, with inversion trapdoors
  - Often more than needed (e.g. SKE needs only PRF)

A Tale of Two Boxes

- Much of today's applied cryptography works with two magic boxes
  - Block Ciphers
  - Hash Functions

- Block Ciphers: Best modeled as (strong) Pseudorandom Permutations, with inversion trapdoors
  - Often more than needed (e.g. SKE needs only PRF)
- Hash Functions:

A Tale of Two Boxes

- Much of today's applied cryptography works with two magic boxes
  - Block Ciphers
  - Hash Functions

- Block Ciphers: Best modeled as (strong) Pseudorandom Permutations, with inversion trapdoors
  - Often more than needed (e.g. SKE needs only PRF)
- Hash Functions:
  - Some times modeled as Random Oracles!

A Tale of Two Boxes

- Much of today's applied cryptography works with two magic boxes
  - Block Ciphers
  - Hash Functions

- Block Ciphers: Best modeled as (strong) Pseudorandom Permutations, with inversion trapdoors
  - Often more than needed (e.g. SKE needs only PRF)
- Hash Functions:
  - Some times modeled as Random Oracles!
    - Schemes relying on this can often be broken

A Tale of Two Boxes

- Much of today's applied cryptography works with two magic boxes
  - Block Ciphers
  - Hash Functions

- Block Ciphers: Best modeled as (strong) Pseudorandom Permutations, with inversion trapdoors
  - Often more than needed (e.g. SKE needs only PRF)
- Hash Functions:
  - Some times modeled as Random Oracles!
    - Schemes relying on this can often be broken
  - Today: understanding security requirements on hash functions

Hash Functions

Hash Functions

- "Randomized" mapping of inputs to shorter hash-values

Hash Functions

- "Randomized" mapping of inputs to shorter hash-values
- Hash functions are useful in various places
  - In data-structures: for efficiency
    - Intuition: hashing removes worst-case effects

Hash Functions

- "Randomized" mapping of inputs to shorter hash-values
- Hash functions are useful in various places
  - In data-structures: for efficiency
    - Intuition: hashing removes worst-case effects
  - In cryptography: for "integrity"

Hash Functions

- "Randomized" mapping of inputs to shorter hash-values
- Hash functions are useful in various places
  - In data-structures: for efficiency
    - Intuition: hashing removes worst-case effects
  - In cryptography: for "integrity"
- Primary use: Domain extension (compress long inputs, and feed them into boxes that can take only short inputs)

Hash Functions

- "Randomized" mapping of inputs to shorter hash-values
- Hash functions are useful in various places
  - In data-structures: for efficiency
    - Intuition: hashing removes worst-case effects
  - In cryptography: for "integrity"
- Primary use: Domain extension (compress long inputs, and feed them into boxes that can take only short inputs)
  - Typical security requirement: "collision resistance"

Hash Functions

- "Randomized" mapping of inputs to shorter hash-values
- Hash functions are useful in various places
  - In data-structures: for efficiency
    - Intuition: hashing removes worst-case effects
  - In cryptography: for "integrity"
- Primary use: Domain extension (compress long inputs, and feed them into boxes that can take only short inputs)
  - Typical security requirement: "collision resistance"
  - Also sometimes: some kind of unpredictability

Hash Function Family

Hash Function Family

- Hash function $h:\{0,1\}^k \rightarrow \{0,1\}^{t(k)}$

Hash Function Family

- Hash function $h:\{0,1\}^k \rightarrow \{0,1\}^{t(k)}$
  - Compresses

Hash Function Family

- Hash function $h:\{0,1\}^k \rightarrow \{0,1\}^{t(k)}$
  - Compresses

x	$h_1(x)$
000	0
001	0
010	0
011	0
100	1
101	1
110	1
111	1

Hash Function Family

- Hash function $h:\{0,1\}^k \rightarrow \{0,1\}^{t(k)}$
  - Compresses
- A family

x	$h_1(x)$	$h_2(x)$	$h_3(x)$	$h_4(x)$	...	$h_N(x)$
000	0	0	0	1		1
001	0	0	1	1		1
010	0	1	0	1		1
011	0	1	1	0		1
100	1	0	0	1		1
101	1	0	1	0		1
110	1	1	0	1		1
111	1	1	1	0		1

Hash Function Family

- Hash function $h:\{0,1\}^k \rightarrow \{0,1\}^{t(k)}$
  - Compresses
- A family
  - Alternately, takes two inputs, the index of the member of the family, and the real input

x	$h_1(x)$	$h_2(x)$	$h_3(x)$	$h_4(x)$	...	$h_N(x)$
000	0	0	0	1		1
001	0	0	1	1		1
010	0	1	0	1		1
011	0	1	1	0		1
100	1	0	0	1		1
101	1	0	1	0		1
110	1	1	0	1		1
111	1	1	1	0		1

Hash Function Family

- Hash function $h:\{0,1\}^k \rightarrow \{0,1\}^{t(k)}$
  - Compresses
- A family
  - Alternately, takes two inputs, the index of the member of the family, and the real input
- Efficient sampling and evaluation

x	$h_1(x)$	$h_2(x)$	$h_3(x)$	$h_4(x)$	...	$h_N(x)$
000	0	0	0	1		1
001	0	0	1	1		1
010	0	1	0	1		1
011	0	1	1	0		1
100	1	0	0	1		1
101	1	0	1	0		1
110	1	1	0	1		1
111	1	1	1	0		1

Hash Function Family

- Hash function $h:\{0,1\}^k \rightarrow \{0,1\}^{t(k)}$
  - Compresses
- A family
  - Alternately, takes two inputs, the index of the member of the family, and the real input
- Efficient sampling and evaluation
- Idea: when the hash function is randomly chosen, "behaves randomly"

x	$h_1(x)$	$h_2(x)$	$h_3(x)$	$h_4(x)$	...	$h_N(x)$
000	0	0	0	1		1
001	0	0	1	1		1
010	0	1	0	1		1
011	0	1	1	0		1
100	1	0	0	1		1
101	1	0	1	0		1
110	1	1	0	1		1
111	1	1	1	0		1

Hash Function Family

- Hash function $h:\{0,1\}^k \rightarrow \{0,1\}^{t(k)}$
  - Compresses
- A family
  - Alternately, takes two inputs, the index of the member of the family, and the real input
- Efficient sampling and evaluation
- Idea: when the hash function is randomly chosen, "behaves randomly"
  - Main goal: to "avoid collisions". Will see several variants of the problem

x	$h_1(x)$	$h_2(x)$	$h_3(x)$	$h_4(x)$	...	$h_N(x)$
000	0	0	0	1		1
001	0	0	1	1		1
010	0	1	0	1		1
011	0	1	1	0		1
100	1	0	0	1		1
101	1	0	1	0		1
110	1	1	0	1		1
111	1	1	1	0		1

Hash Functions in Crypto Practice

Hash Functions in Crypto Practice

- A single fixed function

Hash Functions in Crypto Practice

- A single fixed function
  - e.g. SHA-1, SHA-256, MD4, MD5

Hash Functions in Crypto Practice

- A single fixed function
  - e.g. SHA-1, SHA-256, MD4, MD5
  - Not a family ("unkeyed")

Hash Functions in Crypto Practice

- A single fixed function
  - e.g. SHA-1, SHA-256, MD4, MD5
  - Not a family ("unkeyed")
  - (And no security parameter knob)

Hash Functions in Crypto Practice

- A single fixed function
  - e.g. SHA-1, SHA-256, MD4, MD5
  - Not a family ("unkeyed")
  - (And no security parameter knob)
- Not collision-resistant under any of the following definitions

Hash Functions in Crypto Practice

- A single fixed function
  - e.g. SHA-1, SHA-256, MD4, MD5
  - Not a family ("unkeyed")
  - (And no security parameter knob)
- Not collision-resistant under any of the following definitions
- Alternately, could be considered as have already been randomly chosen from a family (and security parameter fixed too)

Hash Functions in Crypto Practice

- A single fixed function
  - e.g. SHA-1, SHA-256, MD4, MD5
  - Not a family ("unkeyed")
  - (And no security parameter knob)
- Not collision-resistant under any of the following definitions
- Alternately, could be considered as have already been randomly chosen from a family (and security parameter fixed too)
  - Usually involves a "key" (e.g. "I.V.") built into the standard

Degrees of Collision-Resistance

Degrees of Collision-Resistance

- If for all PPT A, Pr[$x \neq y$ and $h(x)=h(y)$] is negligible in the following experiment:

Degrees of Collision-Resistance

- If for all PPT A, Pr[x≠y and h(x)=h(y)] is negligible in the following experiment:
  - A→(x,y); h←$\mathcal{U}$: Combinatorial Hash Functions

Degrees of Collision-Resistance

- If for all PPT A, Pr[x≠y and h(x)=h(y)] is negligible in the following experiment:
  - $A \rightarrow (x,y)$; $h \leftarrow \mathcal{U}$: Combinatorial Hash Functions
  - $A \rightarrow x$; $h \leftarrow \mathcal{U}$; $A(h) \rightarrow y$: Universal One-Way Hash Functions

Degrees of Collision-Resistance

- If for all PPT A, Pr[x≠y and h(x)=h(y)] is negligible in the following experiment:
  - $A \rightarrow (x,y)$; $h \leftarrow \mathcal{H}$: Combinatorial Hash Functions
  - $A \rightarrow x$; $h \leftarrow \mathcal{H}$; $A(h) \rightarrow y$: Universal One-Way Hash Functions
  - $h \leftarrow \mathcal{H}$; $A(h) \rightarrow (x,y)$: Collision-Resistant Hash Functions

Degrees of Collision-Resistance

- If for all PPT A, Pr[x≠y and h(x)=h(y)] is negligible in the following experiment:
  - $A \rightarrow (x,y)$; $h \leftarrow \mathcal{H}$: Combinatorial Hash Functions
  - $A \rightarrow x$; $h \leftarrow \mathcal{H}$; $A(h) \rightarrow y$: Universal One-Way Hash Functions
  - $h \leftarrow \mathcal{H}$; $A(h) \rightarrow (x,y)$: Collision-Resistant Hash Functions
- Also useful sometimes: A gets only oracle access to h(.) (weak). Or, A gets any coins used for sampling h (strong).

Degrees of Collision-Resistance

- If for all PPT A, Pr[x≠y and h(x)=h(y)] is negligible in the following experiment:
  - $A \rightarrow (x,y)$; $h \leftarrow \mathcal{U}$: Combinatorial Hash Functions
  - $A \rightarrow x$; $h \leftarrow \mathcal{U}$; $A(h) \rightarrow y$: Universal One-Way Hash Functions
  - $h \leftarrow \mathcal{U}$; $A(h) \rightarrow (x,y)$: Collision-Resistant Hash Functions
- Also useful sometimes: A gets only oracle access to h(.) (weak). Or, A gets any coins used for sampling h (strong).
- CRHF the strongest; UOWHF still powerful (will be enough for digital signatures)

Degrees of Collision-Resistance

Degrees of Collision-Resistance

- Weaker variants of CRHF (where x is random)

Degrees of Collision-Resistance

- Weaker variants of CRHF (where x is random)
  - $h \leftarrow \mathcal{U}$; $x \leftarrow X$; $A(h,h(x)) \rightarrow y$ (y allowed to be x)

Degrees of Collision-Resistance

- Weaker variants of CRHF (where x is random)
  - $h \leftarrow \mathcal{H}$; $x \leftarrow X$; $A(h,h(x)) \rightarrow y$ (y allowed to be x)
    - Pre-image collision resistance if h(x)=h(y) w.n.p

Degrees of Collision-Resistance

- Weaker variants of CRHF (where x is random)
  - $h \leftarrow \mathcal{U}$; $x \leftarrow X$; $A(h,h(x)) \rightarrow y$ (y allowed to be x)
    - Pre-image collision resistance if h(x)=h(y) w.n.p
    - i.e., f(h,x) := (h,h(x)) is a OWF (and h compresses)

Degrees of Collision-Resistance

- Weaker variants of CRHF (where x is random)
  - $h \leftarrow \mathcal{U}$; $x \leftarrow X$; $A(h,h(x)) \rightarrow y$ (y allowed to be x)
    - Pre-image collision resistance if h(x)=h(y) w.n.p
    - i.e., f(h,x) := (h,h(x)) is a OWF (and h compresses)

A.k.a One-Way Hash Function

Degrees of Collision-Resistance

- Weaker variants of CRHF (where x is random)
  - $h \leftarrow \mathcal{H}$; $x \leftarrow X$; $A(h,h(x)) \rightarrow y$ (y allowed to be x)
    - Pre-image collision resistance if h(x)=h(y) w.n.p
    - i.e., f(h,x) := (h,h(x)) is a OWF (and h compresses)
  - $h \leftarrow \mathcal{H}$; $x \leftarrow X$; $A(h,x) \rightarrow y$ ($y \neq x$)

A.k.a One-Way Hash Function

Degrees of Collision-Resistance

- Weaker variants of CRHF (where x is random)
  - $h \leftarrow \mathcal{H}$; $x \leftarrow X$; $A(h,h(x)) \rightarrow y$ (y allowed to be x)
    - Pre-image collision resistance if h(x)=h(y) w.n.p
    - i.e., f(h,x) := (h,h(x)) is a OWF (and h compresses)
  - $h \leftarrow \mathcal{H}$; $x \leftarrow X$; $A(h,x) \rightarrow y$ ($y \neq x$)
    - Second Pre-image collision resistance if h(x)=h(y) w.n.p

A.k.a One-Way Hash Function

Degrees of Collision-Resistance

- Weaker variants of CRHF (where x is random)
  - $h \leftarrow \mathcal{H}$; $x \leftarrow X$; $A(h,h(x)) \rightarrow y$ (y allowed to be x)
    - Pre-image collision resistance if h(x)=h(y) w.n.p
    - i.e., f(h,x) := (h,h(x)) is a OWF (and h compresses)
  - $h \leftarrow \mathcal{H}$; $x \leftarrow X$; $A(h,x) \rightarrow y$ ($y \neq x$)
    - Second Pre-image collision resistance if h(x)=h(y) w.n.p
  - Incomparable (neither implies the other) [Exercise]

A.k.a One-Way Hash Function

Degrees of Collision-Resistance

- Weaker variants of CRHF (where x is random)
  - $h \leftarrow \mathcal{H}$; $x \leftarrow X$; $A(h,h(x)) \rightarrow y$ (y allowed to be x)
    - Pre-image collision resistance if h(x)=h(y) w.n.p
    - i.e., f(h,x) := (h,h(x)) is a OWF (and h compresses)
  - $h \leftarrow \mathcal{H}$; $x \leftarrow X$; $A(h,x) \rightarrow y$ ($y \neq x$)
    - Second Pre-image collision resistance if h(x)=h(y) w.n.p
  - Incomparable (neither implies the other) [Exercise]
- CRHF implies second pre-image collision resistance and, if sufficiently compressing, then pre-image collision resistance [Exercise]

A.k.a One-Way Hash Function

Hash Length

Hash Length

- If range of the hash function is too small, not collision-resistant

Hash Length

- If range of the hash function is too small, not collision-resistant
  - If range poly-size (i.e. hash log-long), then non-negligible probability that two random x, y provide collision

Hash Length

- If range of the hash function is too small, not collision-resistant
  - If range poly-size (i.e. hash log-long), then non-negligible probability that two random x, y provide collision
- In practice interested in minimizing the hash length (for efficiency)

Hash Length

- If range of the hash function is too small, not collision-resistant
  - If range poly-size (i.e. hash log-long), then non-negligible probability that two random x, y provide collision
- In practice interested in minimizing the hash length (for efficiency)
  - Generic collision-finding attack: birthday attack

Hash Length

- If range of the hash function is too small, not collision-resistant
  - If range poly-size (i.e. hash log-long), then non-negligible probability that two random x, y provide collision
- In practice interested in minimizing the hash length (for efficiency)
  - Generic collision-finding attack: birthday attack
    - Look for a collision in a set of random hashes (needs only oracle access to the hash function)

Hash Length

- If range of the hash function is too small, not collision-resistant
  - If range poly-size (i.e. hash log-long), then non-negligible probability that two random x, y provide collision
- In practice interested in minimizing the hash length (for efficiency)
  - Generic collision-finding attack: birthday attack
    - Look for a collision in a set of random hashes (needs only oracle access to the hash function)
      - Expected size of the set before collision: $O(\sqrt{|range|})$

Hash Length

- If range of the hash function is too small, not collision-resistant
  - If range poly-size (i.e. hash log-long), then non-negligible probability that two random x, y provide collision
- In practice interested in minimizing the hash length (for efficiency)
  - Generic collision-finding attack: birthday attack
    - Look for a collision in a set of random hashes (needs only oracle access to the hash function)
      - Expected size of the set before collision: $O(\sqrt{|range|})$
  - Birthday attack effectively halves the hash length (say security parameter) over "naïve attack"

Universal Hashing

Universal Hashing

- Combinatorial HF: A→(x,y); h←ℋ. h(x)=h(y) w.n.p

Universal Hashing

- Combinatorial HF: A→(x,y); h←ℋ. h(x)=h(y) w.n.p
- Even better: 2-Universal Hash Functions

Universal Hashing

- Combinatorial HF: A→(x,y); h←ℋ. h(x)=h(y) w.n.p
- Even better: 2-Universal Hash Functions
  - "Uniform" and "Pairwise-independent"

Universal Hashing

- Combinatorial HF: $A \to (x,y)$; $h \leftarrow_{\$}$. $h(x)=h(y)$ w.n.p
- Even better: 2-Universal Hash Functions
  - "Uniform" and "Pairwise-independent"
  - $\forall x,z \; Pr_{h \leftarrow_{\$}} [\, h(x)=z \,] = 1/|Z|$ (where $h:X \to Z$)

Universal Hashing

- Combinatorial HF: $A \to (x,y)$; $h \leftarrow \mathcal{U}$. $h(x)=h(y)$ w.n.p
- Even better: 2-Universal Hash Functions
  - "Uniform" and "Pairwise-independent"
  - $\forall x,z \ Pr_{h \leftarrow \mathcal{U}}[\ h(x)=z\ ] = 1/|Z|$ (where $h:X \to Z$)

x	$h_1(x)$	$h_2(x)$	$h_3(x)$	$h_4(x)$
0	0	0	1	1
1	0	1	0	1
2	1	0	0	1

Universal Hashing

- Combinatorial HF: $A \to (x,y)$; $h \leftarrow \mathcal{U}$. $h(x)=h(y)$ w.n.p
- Even better: 2-Universal Hash Functions
  - "Uniform" and "Pairwise-independent"
  - $\forall x,z \ \Pr_{h \leftarrow \mathcal{U}}[\ h(x)=z\ ] = 1/|Z|$ (where $h:X \to Z$)
  - $\forall x \neq y,w,z \ \Pr_{h \leftarrow \mathcal{U}}[\ h(x)=w,\ h(y)=z\ ] = 1/|Z|^2$

x	$h_1(x)$	$h_2(x)$	$h_3(x)$	$h_4(x)$
0	0	0	1	1
1	0	1	0	1
2	1	0	0	1

Universal Hashing

- Combinatorial HF: $A \to (x,y)$; $h \leftarrow \mathcal{U}$. $h(x)=h(y)$ w.n.p
- Even better: 2-Universal Hash Functions
  - "Uniform" and "Pairwise-independent"
  - $\forall x,z \ Pr_{h \leftarrow \mathcal{U}}[\ h(x)=z\ ] = 1/|Z|$ (where $h:X \to Z$)
  - $\forall x \neq y,w,z \ Pr_{h \leftarrow \mathcal{U}}[\ h(x)=w,\ h(y)=z\ ] = 1/|Z|^2$
    - $\forall x \neq y \ Pr_{h \leftarrow \mathcal{U}}[\ h(x)=h(y)\ ] = 1/|Z|$

x	$h_1(x)$	$h_2(x)$	$h_3(x)$	$h_4(x)$
0	0	0	1	1
1	0	1	0	1
2	1	0	0	1

Universal Hashing

- Combinatorial HF: $A \to (x,y)$; $h \leftarrow \mathcal{U}$. $h(x)=h(y)$ w.n.p
- Even better: 2-Universal Hash Functions
  - "Uniform" and "Pairwise-independent"
  - $\forall x,z \ Pr_{h \leftarrow \mathcal{U}}[\ h(x)=z\ ] = 1/|Z|$ (where $h:X \to Z$)
  - $\forall x \neq y,w,z \ Pr_{h \leftarrow \mathcal{U}}[\ h(x)=w,\ h(y)=z\ ] = 1/|Z|^2$
    - $\forall x \neq y \ Pr_{h \leftarrow \mathcal{U}}[\ h(x)=h(y)\ ] = 1/|Z|$

x	$h_1(x)$	$h_2(x)$	$h_3(x)$	$h_4(x)$
0	0	0	1	1
1	0	1	0	1
2	1	0	0	1

Negligible collision-probability if super-polynomial-sized range

Universal Hashing

- Combinatorial HF: $A \to (x,y)$; $h \leftarrow \mathcal{U}$. $h(x)=h(y)$ w.n.p
- Even better: 2-Universal Hash Functions
  - "Uniform" and "Pairwise-independent"
  - $\forall x,z \ Pr_{h \leftarrow \mathcal{U}}[\ h(x)=z\ ] = 1/|Z|$ (where $h:X \to Z$)
  - $\forall x \neq y,w,z \ Pr_{h \leftarrow \mathcal{U}}[\ h(x)=w,\ h(y)=z\ ] = 1/|Z|^2$
    - $\forall x \neq y \ Pr_{h \leftarrow \mathcal{U}}[\ h(x)=h(y)\ ] = 1/|Z|$
- k-Universal:

x	$h_1(x)$	$h_2(x)$	$h_3(x)$	$h_4(x)$
0	0	0	1	1
1	0	1	0	1
2	1	0	0	1

Negligible collision-probability if super-polynomial-sized range

Universal Hashing

- Combinatorial HF: $A \to (x,y)$; $h \leftarrow \mathcal{U}$. $h(x)=h(y)$ w.n.p
- Even better: 2-Universal Hash Functions
  - "Uniform" and "Pairwise-independent"
  - $\forall x,z \ Pr_{h \leftarrow \mathcal{U}}[\ h(x)=z\ ] = 1/|Z|$ (where $h:X \to Z$)
  - $\forall x \neq y,w,z \ Pr_{h \leftarrow \mathcal{U}}[\ h(x)=w,\ h(y)=z\ ] = 1/|Z|^2$
    - $\forall x \neq y \ Pr_{h \leftarrow \mathcal{U}}[\ h(x)=h(y)\ ] = 1/|Z|$
- k-Universal:
  - $\forall x_1..x_k \ z_1..z_k \ Pr_{h \leftarrow \mathcal{U}}[\forall i \ h(x_i)=z_i\ ] = 1/|Z|^k$

x	$h_1(x)$	$h_2(x)$	$h_3(x)$	$h_4(x)$
0	0	0	1	1
1	0	1	0	1
2	1	0	0	1

Negligible collision-probability if super-polynomial-sized range

Universal Hashing

- Combinatorial HF: $A \to (x,y)$; $h \leftarrow \mathcal{H}$. $h(x)=h(y)$ w.n.p
- Even better: 2-Universal Hash Functions
  - "Uniform" and "Pairwise-independent"
  - $\forall x,z\ Pr_{h \leftarrow \mathcal{H}}[\ h(x)=z\ ] = 1/|Z|$ (where $h:X \to Z$)
  - $\forall x \neq y,w,z\ Pr_{h \leftarrow \mathcal{H}}[\ h(x)=w,\ h(y)=z\ ] = 1/|Z|^2$
    - $\forall x \neq y\ Pr_{h \leftarrow \mathcal{H}}[\ h(x)=h(y)\ ] = 1/|Z|$
- k-Universal:
  - $\forall x_1..x_k\ z_1..z_k\ Pr_{h \leftarrow \mathcal{H}}[\forall i\ h(x_i)=z_i\ ] = 1/|Z|^k$
- Inefficient example: $\mathcal{H}$ set of all functions from X to Z

x	$h_1(x)$	$h_2(x)$	$h_3(x)$	$h_4(x)$
0	0	0	1	1
1	0	1	0	1
2	1	0	0	1

Negligible collision-probability if super-polynomial-sized range

Universal Hashing

- Combinatorial HF: $A \to (x,y)$; $h \leftarrow \mathcal{H}$. $h(x)=h(y)$ w.n.p
- Even better: 2-Universal Hash Functions
  - "Uniform" and "Pairwise-independent"
  - $\forall x,z\ \Pr_{h \leftarrow \mathcal{H}}[\ h(x)=z\ ] = 1/|Z|$ (where $h:X \to Z$)
  - $\forall x \neq y,w,z\ \Pr_{h \leftarrow \mathcal{H}}[\ h(x)=w,\ h(y)=z\ ] = 1/|Z|^2$
    - $\forall x \neq y\ \Pr_{h \leftarrow \mathcal{H}}[\ h(x)=h(y)\ ] = 1/|Z|$
- k-Universal:
  - $\forall x_1..x_k\ z_1..z_k\ \Pr_{h \leftarrow \mathcal{H}}[\forall i\ h(x_i)=z_i\ ] = 1/|Z|^k$
- Inefficient example: $\mathcal{H}$ set of all functions from X to Z
  - But we will need all $h \in \mathcal{H}$ to be succinctly described and efficiently evaluable

x	$h_1(x)$	$h_2(x)$	$h_3(x)$	$h_4(x)$
0	0	0	1	1
1	0	1	0	1
2	1	0	0	1

Negligible collision-probability if super-polynomial-sized range

Universal Hashing

- Combinatorial HF: $A \to (x,y)$; $h \leftarrow \mathcal{U}$. $h(x)=h(y)$ w.n.p
- Even better: 2-Universal Hash Functions
  - "Uniform" and "Pairwise-independent"
  - $\forall x,z \ \Pr_{h \leftarrow \mathcal{U}}[\ h(x)=z\ ] = 1/|Z|$ (where $h:X \to Z$)
  - $\forall x \neq y,w,z \ \Pr_{h \leftarrow \mathcal{U}}[\ h(x)=w,\ h(y)=z\ ] = 1/|Z|^2$
    - $\forall x \neq y \ \Pr_{h \leftarrow \mathcal{U}}[\ h(x)=h(y)\ ] = 1/|Z|$

x	$h_1(x)$	$h_2(x)$	$h_3(x)$	$h_4(x)$
0	0	0	1	1
1	0	1	0	1
2	1	0	0	1

Negligible collision-probability if super-polynomial-sized range

Universal Hashing

- Combinatorial HF: $A \to (x,y)$; $h \leftarrow \mathcal{U}$. $h(x)=h(y)$ w.n.p
- Even better: 2-Universal Hash Functions
  - "Uniform" and "Pairwise-independent"
  - $\forall x,z \ Pr_{h \leftarrow \mathcal{U}}[\ h(x)=z\ ] = 1/|Z|$ (where $h:X \to Z$)
  - $\forall x \neq y,w,z \ Pr_{h \leftarrow \mathcal{U}}[\ h(x)=w,\ h(y)=z\ ] = 1/|Z|^2$
    - $\forall x \neq y \ Pr_{h \leftarrow \mathcal{U}}[\ h(x)=h(y)\ ] = 1/|Z|$
- e.g. $h_{a,b}(x) = ax+b$ (in a finite field, $X=Z$)

x	$h_1(x)$	$h_2(x)$	$h_3(x)$	$h_4(x)$
0	0	0	1	1
1	0	1	0	1
2	1	0	0	1

Negligible collision-probability if super-polynomial-sized range

Universal Hashing

- Combinatorial HF: $A \to (x,y)$; $h \leftarrow \mathcal{H}$. $h(x)=h(y)$ w.n.p
- Even better: 2-Universal Hash Functions
  - "Uniform" and "Pairwise-independent"
  - $\forall x,z \ \Pr_{h \leftarrow \mathcal{H}}[\ h(x)=z\ ] = 1/|Z|$ (where $h:X \to Z$)
  - $\forall x \neq y,w,z \ \Pr_{h \leftarrow \mathcal{H}}[\ h(x)=w,\ h(y)=z\ ] = 1/|Z|^2$
    - $\forall x \neq y \ \Pr_{h \leftarrow \mathcal{H}}[\ h(x)=h(y)\ ] = 1/|Z|$
- e.g. $h_{a,b}(x) = ax+b$ (in a finite field, $X=Z$)
  - $\Pr_{a,b}[\ ax+b = z\ ] = \Pr_{a,b}[\ b = z-ax\ ] = 1/|Z|$

x	$h_1(x)$	$h_2(x)$	$h_3(x)$	$h_4(x)$
0	0	0	1	1
1	0	1	0	1
2	1	0	0	1

Negligible collision-probability if super-polynomial-sized range

Universal Hashing

- Combinatorial HF: $A \to (x,y)$; $h \leftarrow \mathcal{H}$. $h(x)=h(y)$ w.n.p
- Even better: 2-Universal Hash Functions
  - "Uniform" and "Pairwise-independent"
  - $\forall x,z \ \Pr_{h \leftarrow \mathcal{H}}[\ h(x)=z\ ] = 1/|Z|$ (where $h:X \to Z$)
  - $\forall x \neq y,w,z \ \Pr_{h \leftarrow \mathcal{H}}[\ h(x)=w,\ h(y)=z\ ] = 1/|Z|^2$
    - $\forall x \neq y \ \Pr_{h \leftarrow \mathcal{H}}[\ h(x)=h(y)\ ] = 1/|Z|$
- e.g. $h_{a,b}(x) = ax+b$ (in a finite field, $X=Z$)
  - $\Pr_{a,b}[\ ax+b = z\ ] = \Pr_{a,b}[\ b = z-ax\ ] = 1/|Z|$
  - $\Pr_{a,b}[\ ax+b = w,\ ay+b = z] = ?$ Exactly one (a,b) satisfying the two equations (for $x \neq y$)

x	$h_1(x)$	$h_2(x)$	$h_3(x)$	$h_4(x)$
0	0	0	1	1
1	0	1	0	1
2	1	0	0	1

Negligible collision-probability if super-polynomial-sized range

Universal Hashing

- Combinatorial HF: $A \to (x,y)$; $h \leftarrow \mathcal{H}$. $h(x)=h(y)$ w.n.p
- Even better: 2-Universal Hash Functions
  - "Uniform" and "Pairwise-independent"
  - $\forall x,z \ \Pr_{h \leftarrow \mathcal{H}}[\ h(x)=z\ ] = 1/|Z|$ (where $h: X \to Z$)
  - $\forall x \neq y,w,z \ \Pr_{h \leftarrow \mathcal{H}}[\ h(x)=w,\ h(y)=z\ ] = 1/|Z|^2$
    - $\forall x \neq y \ \Pr_{h \leftarrow \mathcal{H}}[\ h(x)=h(y)\ ] = 1/|Z|$
- e.g. $h_{a,b}(x) = ax+b$ (in a finite field, $X=Z$)
  - $\Pr_{a,b}[\ ax+b = z\ ] = \Pr_{a,b}[\ b = z-ax\ ] = 1/|Z|$
  - $\Pr_{a,b}[\ ax+b = w,\ ay+b = z] = ?$ Exactly one (a,b) satisfying the two equations (for $x \neq y$)
    - $\Pr_{a,b}[\ ax+b = w,\ ay+b = z] = 1/|Z|^2$

x	$h_1(x)$	$h_2(x)$	$h_3(x)$	$h_4(x)$
0	0	0	1	1
1	0	1	0	1
2	1	0	0	1

Negligible collision-probability if super-polynomial-sized range

Universal Hashing

- Combinatorial HF: $A \to (x,y)$; $h \leftarrow \mathcal{U}$. $h(x)=h(y)$ w.n.p
- Even better: 2-Universal Hash Functions
  - "Uniform" and "Pairwise-independent"
  - $\forall x,z \ \Pr_{h \leftarrow \mathcal{U}}[\ h(x)=z\ ] = 1/|Z|$ (where $h: X \to Z$)
  - $\forall x \neq y,w,z \ \Pr_{h \leftarrow \mathcal{U}}[\ h(x)=w,\ h(y)=z\ ] = 1/|Z|^2$
    - $\forall x \neq y \ \Pr_{h \leftarrow \mathcal{U}}[\ h(x)=h(y)\ ] = 1/|Z|$
- e.g. $h_{a,b}(x) = ax+b$ (in a finite field, $X=Z$)
  - $\Pr_{a,b}[\ ax+b = z\ ] = \Pr_{a,b}[\ b = z-ax\ ] = 1/|Z|$
  - $\Pr_{a,b}[\ ax+b = w,\ ay+b = z] = ?$ Exactly one (a,b) satisfying the two equations (for $x \neq y$)
    - $\Pr_{a,b}[\ ax+b = w,\ ay+b = z] = 1/|Z|^2$
  - But does not compress!

x	$h_1(x)$	$h_2(x)$	$h_3(x)$	$h_4(x)$
0	0	0	1	1
1	0	1	0	1
2	1	0	0	1

Negligible collision-probability if super-polynomial-sized range

Universal Hashing

- Combinatorial HF: $A \to (x,y)$; $h \leftarrow \mathcal{U}$. $h(x)=h(y)$ w.n.p
- Even better: 2-Universal Hash Functions
  - "Uniform" and "Pairwise-independent"
  - $\forall x,z \ \Pr_{h \leftarrow \mathcal{U}}[\ h(x)=z\ ] = 1/|Z|$ (where $h:X \to Z$)
  - $\forall x \neq y,w,z \ \Pr_{h \leftarrow \mathcal{U}}[\ h(x)=w,\ h(y)=z\ ] = 1/|Z|^2$
    - $\forall x \neq y \ \Pr_{h \leftarrow \mathcal{U}}[\ h(x)=h(y)\ ] = 1/|Z|$

x	$h_1(x)$	$h_2(x)$	$h_3(x)$	$h_4(x)$
0	0	0	1	1
1	0	1	0	1
2	1	0	0	1

Negligible collision-probability if super-polynomial-sized range

Universal Hashing

- Combinatorial HF: $A \to (x,y)$; $h \leftarrow \mathcal{U}$. $h(x)=h(y)$ w.n.p
- Even better: 2-Universal Hash Functions
  - "Uniform" and "Pairwise-independent"
  - $\forall x,z \ Pr_{h \leftarrow \mathcal{U}}[\ h(x)=z\ ] = 1/|Z|$ (where $h:X \to Z$)
  - $\forall x \neq y,w,z \ Pr_{h \leftarrow \mathcal{U}}[\ h(x)=w,\ h(y)=z\ ] = 1/|Z|^2$
    - $\forall x \neq y \ Pr_{h \leftarrow \mathcal{U}}[\ h(x)=h(y)\ ] = 1/|Z|$
- e.g. $h'_h(x)$ = Chop(h(x)) where h from a (possibly non-compressing) 2-universal HF

x	$h_1(x)$	$h_2(x)$	$h_3(x)$	$h_4(x)$
0	0	0	1	1
1	0	1	0	1
2	1	0	0	1

Negligible collision-probability if super-polynomial-sized range

Universal Hashing

- Combinatorial HF: $A \to (x,y)$; $h \leftarrow \mathcal{U}$. $h(x)=h(y)$ w.n.p
- Even better: 2-Universal Hash Functions
  - "Uniform" and "Pairwise-independent"
  - $\forall x,z \ Pr_{h \leftarrow \mathcal{U}}[\ h(x)=z\ ] = 1/|Z|$ (where $h:X \to Z$)
  - $\forall x \neq y,w,z \ Pr_{h \leftarrow \mathcal{U}}[\ h(x)=w,\ h(y)=z\ ] = 1/|Z|^2$
    - $\forall x \neq y \ Pr_{h \leftarrow \mathcal{U}}[\ h(x)=h(y)\ ] = 1/|Z|$
- e.g. $h'_h(x)$ = Chop(h(x)) where h from a (possibly non-compressing) 2-universal HF
  - Chop a t-to-1 map (e.g. removes last bit: 2-to-1)

x	$h_1(x)$	$h_2(x)$	$h_3(x)$	$h_4(x)$
0	0	0	1	1
1	0	1	0	1
2	1	0	0	1

Negligible collision-probability if super-polynomial-sized range

Universal Hashing

- Combinatorial HF: $A \to (x,y)$; $h \leftarrow \mathcal{U}$. $h(x)=h(y)$ w.n.p
- Even better: 2-Universal Hash Functions
  - "Uniform" and "Pairwise-independent"
  - $\forall x,z \ \Pr_{h \leftarrow \mathcal{U}}[\ h(x)=z\ ] = 1/|Z|$ (where $h:X \to Z$)
  - $\forall x \neq y,w,z \ \Pr_{h \leftarrow \mathcal{U}}[\ h(x)=w,\ h(y)=z\ ] = 1/|Z|^2$
    - $\forall x \neq y \ \Pr_{h \leftarrow \mathcal{U}}[\ h(x)=h(y)\ ] = 1/|Z|$

x	$h_1(x)$	$h_2(x)$	$h_3(x)$	$h_4(x)$
0	0	0	1	1
1	0	1	0	1
2	1	0	0	1

Negligible collision-probability if super-polynomial-sized range

- e.g. $h'_h(x)$ = Chop(h(x)) where h from a (possibly non-compressing) 2-universal HF
  - Chop a t-to-1 map (e.g. removes last bit: 2-to-1)
    - $\Pr_h[\ \text{Chop}(h(x)) = w,\ \text{Chop}(h(y)) = z]$
      $= \Pr_h[\ h(x) = w0 \text{ or } w1,\ h(y) = z0 \text{ or } z1] = 4/|Z|^2$

UOWHF

UOWHF

- Universal One-Way HF: $A \rightarrow x$; $h \leftarrow \mathcal{U}$; $A(h) \rightarrow y$. $h(x)=h(y)$ w.n.p

UOWHF

- Universal One-Way HF: $A \rightarrow x$; $h \leftarrow \mathcal{U}$; $A(h) \rightarrow y$. $h(x)=h(y)$ w.n.p
- Can be constructed from OWF

UOWHF

- Universal One-Way HF: $A \rightarrow x$; $h \leftarrow \mathcal{U}$; $A(h) \rightarrow y$. $h(x)=h(y)$ w.n.p
- Can be constructed from OWF
- Easier to see OWP $\Rightarrow$ UOWHF

UOWHF

- Universal One-Way HF: $A \rightarrow x$; $h \leftarrow \mathcal{H}$; $A(h) \rightarrow y$. $h(x)=h(y)$ w.n.p
- Can be constructed from OWF
- Easier to see OWP $\Rightarrow$ UOWHF
  - $F_h(x) = h(f(x))$, where f is a OWP and h from a UHF family

UOWHF

- Universal One-Way HF: $A \rightarrow x$; $h \leftarrow \mathcal{H}$; $A(h) \rightarrow y$. $h(x)=h(y)$ w.n.p
- Can be constructed from OWF
- Easier to see OWP $\Rightarrow$ UOWHF
  - $F_h(x) = h(f(x))$, where f is a OWP and h from a UHF family
    - h compresses by a bit (i.e., 2-to-1 maps), and

UOWHF

- Universal One-Way HF: $A \rightarrow x$; $h \leftarrow \mathcal{U}$; $A(h) \rightarrow y$. $h(x)=h(y)$ w.n.p
- Can be constructed from OWF
- Easier to see OWP $\Rightarrow$ UOWHF
  - $F_h(x) = h(f(x))$, where f is a OWP and h from a UHF family
    - h compresses by a bit (i.e., 2-to-1 maps), and
    - for all z,z', can sample (solve for) h s.t. $h(z) = h(z')$

UOWHF

- Universal One-Way HF: $A \rightarrow x$; $h \leftarrow \mathcal{U}$; $A(h) \rightarrow y$. $h(x)=h(y)$ w.n.p
- Can be constructed from OWF
- Easier to see OWP $\Rightarrow$ UOWHF
  - $F_h(x) = h(f(x))$, where f is a OWP and h from a UHF family
    - h compresses by a bit (i.e., 2-to-1 maps), and
    - for all z,z', can sample (solve for) h s.t. $h(z) = h(z')$
  - Is a UOWHF [Why?]

UOWHF

- Universal One-Way HF: $A \rightarrow x$; $h \leftarrow \mathcal{U}$; $A(h) \rightarrow y$. $h(x)=h(y)$ w.n.p
- Can be constructed from OWF
- Easier to see OWP $\Rightarrow$ UOWHF
  - $F_h(x) = h(f(x))$, where f is a OWP and h from a UHF family
    - h compresses by a bit (i.e., 2-to-1 maps), and
    - for all z,z', can sample (solve for) h s.t. $h(z) = h(z')$
  - Is a UOWHF [Why?]
  - Gives a UOWHF with range and domain same as the UHF

UOWHF

- Universal One-Way HF: $A \rightarrow x$; $h \leftarrow \mathcal{U}$; $A(h) \rightarrow y$. $h(x)=h(y)$ w.n.p
- Can be constructed from OWF
- Easier to see OWP $\Rightarrow$ UOWHF
  - $F_h(x) = h(f(x))$, where f is a OWP and h from a UHF family
    - h compresses by a bit (i.e., 2-to-1 maps), and
    - for all z,z', can sample (solve for) h s.t. $h(z) = h(z')$
  - Is a UOWHF [Why?]
  - Gives a UOWHF with range and domain same as the UHF
    - Will see shortly, how to extend the domain to arbitrarily long strings (without increasing output size)

CRHF

CRHF

- Collision-Resistant HF: $h \leftarrow \mathcal{H}$; $A(h) \rightarrow (x,y)$. $h(x)=h(y)$ w.n.p

CRHF

- Collision-Resistant HF: $h \leftarrow \mathcal{K}$; $A(h) \rightarrow (x,y)$. $h(x)=h(y)$ w.n.p
- Not known to be possible from OWF/OWP alone

CRHF

- Collision-Resistant HF: $h \leftarrow \mathcal{U}$; $A(h) \rightarrow (x,y)$. $h(x)=h(y)$ w.n.p
- Not known to be possible from OWF/OWP alone
  - "Impossibility" (blackbox-separation) known

CRHF

- Collision-Resistant HF: $h \leftarrow \mathcal{U}$; $A(h) \rightarrow (x,y)$. $h(x)=h(y)$ w.n.p
- Not known to be possible from OWF/OWP alone
  - "Impossibility" (blackbox-separation) known
- Possible from "claw-free pair of permutations"

CRHF

- Collision-Resistant HF: $h \leftarrow \mathcal{K}$; $A(h) \rightarrow (x,y)$. $h(x)=h(y)$ w.n.p
- Not known to be possible from OWF/OWP alone
  - "Impossibility" (blackbox-separation) known
- Possible from "claw-free pair of permutations"
  - In turn from hardness of discrete-log, factoring, and from lattice-based assumptions

CRHF

- Collision-Resistant HF: $h \leftarrow \mathcal{K}$; $A(h) \rightarrow (x,y)$. $h(x)=h(y)$ w.n.p
- Not known to be possible from OWF/OWP alone
  - "Impossibility" (blackbox-separation) known
- Possible from "claw-free pair of permutations"
  - In turn from hardness of discrete-log, factoring, and from lattice-based assumptions
- Also from "homomorphic one-way permutations", and from homomorphic encryptions

CRHF

- Collision-Resistant HF: $h \leftarrow \mathcal{K}$; $A(h) \rightarrow (x,y)$. $h(x)=h(y)$ w.n.p
- Not known to be possible from OWF/OWP alone
  - "Impossibility" (blackbox-separation) known
- Possible from "claw-free pair of permutations"
  - In turn from hardness of discrete-log, factoring, and from lattice-based assumptions
- Also from "homomorphic one-way permutations", and from homomorphic encryptions
  - All candidates use mathematical structures that are considered computationally expensive

CRHF

CRHF

- CRHF from discrete log assumption:

CRHF

- CRHF from discrete log assumption:
  - Suppose $\mathbb{G}$ a group of prime order q, where DL is considered hard (e.g. $\mathbb{QR}_p^*$ for p=2q+1 a safe prime)

CRHF

- CRHF from discrete log assumption:
  - Suppose $\mathbb{G}$ a group of prime order q, where DL is considered hard (e.g. $\mathbb{QR}_p^*$ for p=2q+1 a safe prime)
  - $h_{g1,g2}(x_1,x_2) = g_1^{x1} g_2^{x2}$ (in $\mathbb{G}$) where g_1, $g_2 \neq 1$ (hence generators)

CRHF

- CRHF from discrete log assumption:
  - Suppose $\mathbb{G}$ a group of prime order q, where DL is considered hard (e.g. $\mathbb{QR}_p^*$ for p=2q+1 a safe prime)
  - $h_{g1,g2}(x_1,x_2) = g_1^{x1} g_2^{x2}$ (in $\mathbb{G}$) where g_1, $g_2 \neq 1$ (hence generators)
  - A collision: $(x_1,x_2) \neq (y_1,y_2)$ s.t. $h_{g1,g2}(x_1,x_2) = h_{g1,g2}(y_1,y_2)$

CRHF

- CRHF from discrete log assumption:
  - Suppose $\mathbb{G}$ a group of prime order q, where DL is considered hard (e.g. $\mathbb{QR}_p^*$ for p=2q+1 a safe prime)
  - $h_{g1,g2}(x_1,x_2) = g_1^{x1} g_2^{x2}$ (in $\mathbb{G}$) where g_1, $g_2 \neq 1$ (hence generators)
  - A collision: $(x_1,x_2) \neq (y_1,y_2)$ s.t. $h_{g1,g2}(x_1,x_2) = h_{g1,g2}(y_1,y_2)$
    - Then $(x_1,x_2) \neq (y_1,y_2) \Rightarrow x_1 \neq y_1$ and $x_2 \neq y_2$ [Why?]

CRHF

- CRHF from discrete log assumption:
  - Suppose $\mathbb{G}$ a group of prime order q, where DL is considered hard (e.g. $\mathbb{QR}_p^*$ for p=2q+1 a safe prime)
  - $h_{g1,g2}(x_1,x_2) = g_1^{x1} g_2^{x2}$ (in $\mathbb{G}$) where g_1, $g_2 \neq 1$ (hence generators)
  - A collision: $(x_1,x_2) \neq (y_1,y_2)$ s.t. $h_{g1,g2}(x_1,x_2) = h_{g1,g2}(y_1,y_2)$
    - Then $(x_1,x_2) \neq (y_1,y_2) \Rightarrow x_1 \neq y_1$ and $x_2 \neq y_2$ [Why?]
    - Then $g_2 = g_1^{(x1-y1)/(x2-y2)}$ (exponents in $\mathbb{Z}_q^*$)

CRHF

- CRHF from discrete log assumption:
  - Suppose $\mathbb{G}$ a group of prime order q, where DL is considered hard (e.g. $\mathbb{QR}_p^*$ for p=2q+1 a safe prime)
  - $h_{g1,g2}(x_1,x_2) = g_1^{x1} g_2^{x2}$ (in $\mathbb{G}$) where g_1, $g_2 \neq 1$ (hence generators)
  - A collision: $(x_1,x_2) \neq (y_1,y_2)$ s.t. $h_{g1,g2}(x_1,x_2) = h_{g1,g2}(y_1,y_2)$
    - Then $(x_1,x_2) \neq (y_1,y_2) \Rightarrow x_1 \neq y_1$ and $x_2 \neq y_2$ [Why?]
    - Then $g_2 = g_1^{(x1-y1)/(x2-y2)}$ (exponents in $\mathbb{Z}_q^*$)
      - i.e., for some base g_1, can compute DL of g_2 (a random non-unit element)

CRHF

- CRHF from discrete log assumption:
  - Suppose $\mathbb{G}$ a group of prime order q, where DL is considered hard (e.g. $\mathbb{QR}_p^*$ for p=2q+1 a safe prime)
  - $h_{g1,g2}(x_1,x_2) = g_1^{x1} g_2^{x2}$ (in $\mathbb{G}$) where g_1, $g_2 \neq 1$ (hence generators)
  - A collision: $(x_1,x_2) \neq (y_1,y_2)$ s.t. $h_{g1,g2}(x_1,x_2) = h_{g1,g2}(y_1,y_2)$
    - Then $(x_1,x_2) \neq (y_1,y_2) \Rightarrow x_1 \neq y_1$ and $x_2 \neq y_2$ [Why?]
    - Then $g_2 = g_1^{(x1-y1)/(x2-y2)}$ (exponents in $\mathbb{Z}_q^*$)
      - i.e., for some base g_1, can compute DL of g_2 (a random non-unit element)
  - Hash halves the size of the input

Domain Extension

Domain Extension

- Full-domain hash: hash arbitrarily long strings to a single hash value

Domain Extension

- Full-domain hash: hash arbitrarily long strings to a single hash value
  - So far, UOWHF/CRHF which have a fixed domain

Domain Extension

- Full-domain hash: hash arbitrarily long strings to a single hash value
  - So far, UOWHF/CRHF which have a fixed domain
- Idea 1: by repeated application

Domain Extension

- Full-domain hash: hash arbitrarily long strings to a single hash value
  - So far, UOWHF/CRHF which have a fixed domain
- Idea 1: by repeated application
  - If one-bit compression, to hash n-bit string, O(n) (independent) invocations of the basic hash function

Domain Extension

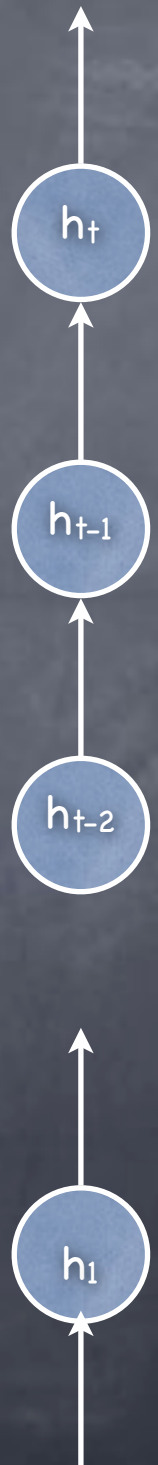
- Full-domain hash: hash arbitrarily long strings to a single hash value
  - So far, UOWHF/CRHF which have a fixed domain
- Idea 1: by repeated application
  - If one-bit compression, to hash n-bit string, O(n) (independent) invocations of the basic hash function

Domain Extension

- Full-domain hash: hash arbitrarily long strings to a single hash value
  - So far, UOWHF/CRHF which have a fixed domain
- Idea 1: by repeated application
  - If one-bit compression, to hash n-bit string, O(n) (independent) invocations of the basic hash function
    - Independent invocations: hash description depends on n (linearly)

Domain Extension

- Full-domain hash: hash arbitrarily long strings to a single hash value
  - So far, UOWHF/CRHF which have a fixed domain
- Idea 1: by repeated application
  - If one-bit compression, to hash n-bit string, O(n) (independent) invocations of the basic hash function
    - Independent invocations: hash description depends on n (linearly)



Domain Extension

Domain Extension

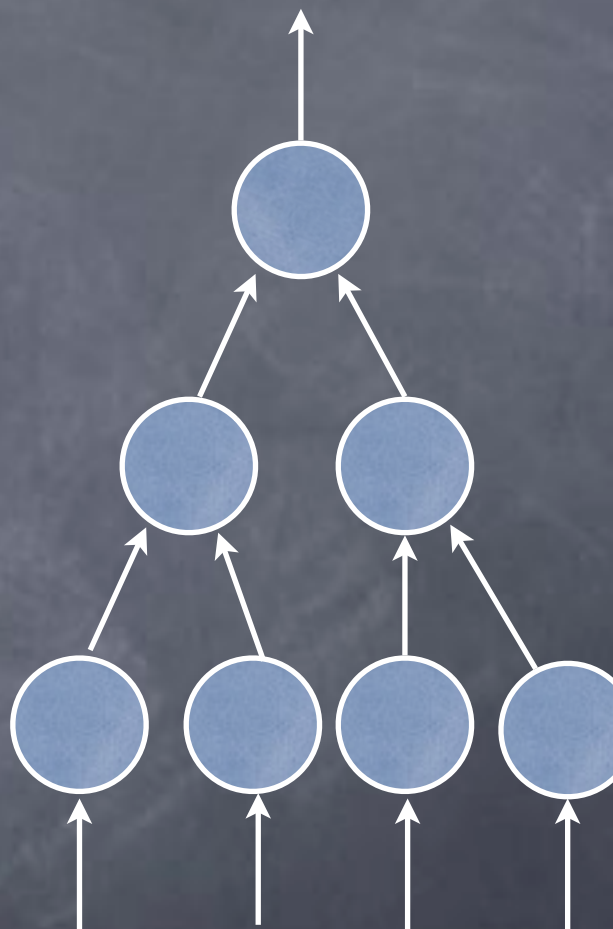
- Can compose hash functions more efficiently, using a "Merkle tree"

Domain Extension

- Can compose hash functions more efficiently, using a "Merkle tree"
  - Suppose basic hash from $\{0,1\}^k$ to $\{0,1\}^{k/2}$. A hash function from $\{0,1\}^{4k}$ to $\{0,1\}^{k/2}$ using a tree of depth 3

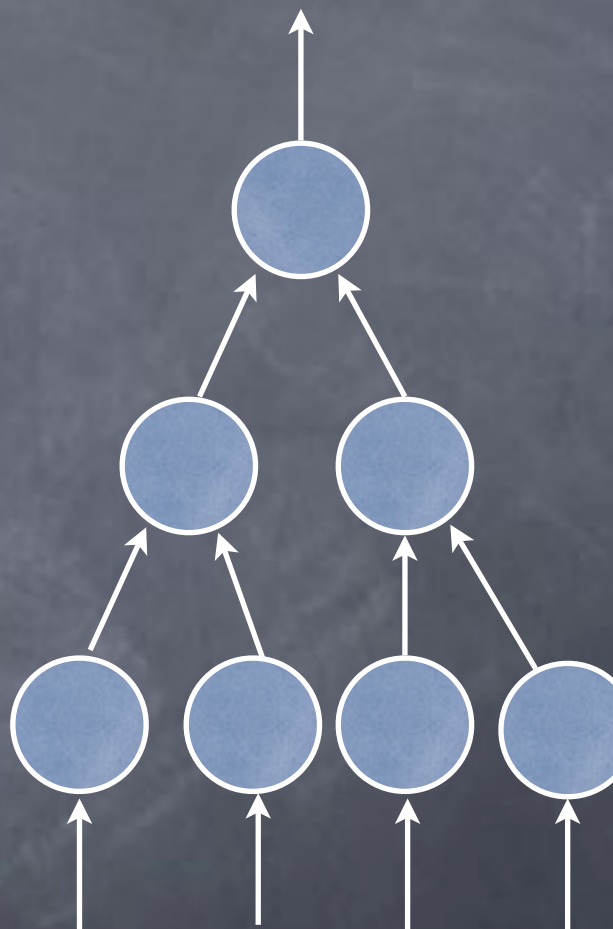
Domain Extension

- Can compose hash functions more efficiently, using a "Merkle tree"
  - Suppose basic hash from $\{0,1\}^k$ to $\{0,1\}^{k/2}$. A hash function from $\{0,1\}^{4k}$ to $\{0,1\}^{k/2}$ using a tree of depth 3



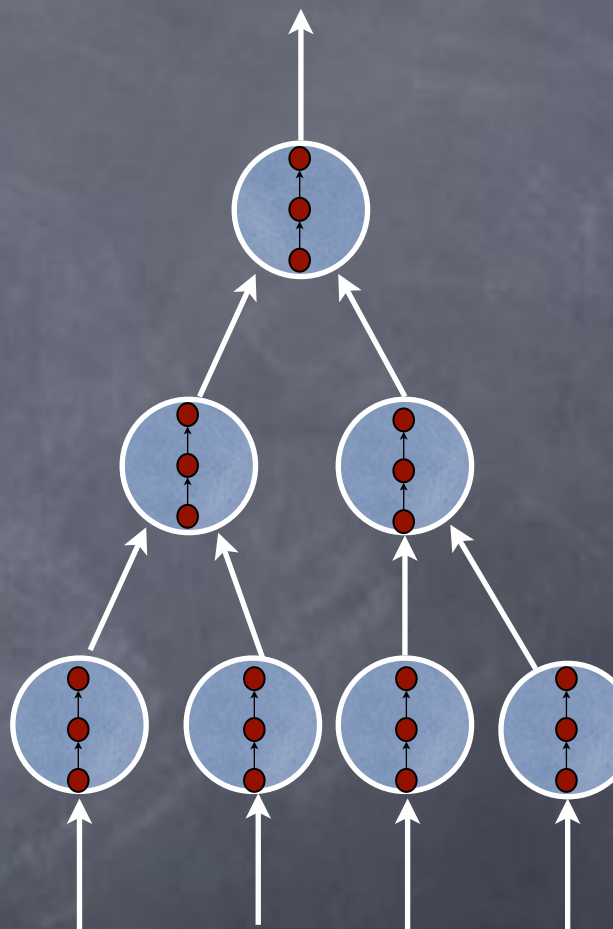
Domain Extension

- Can compose hash functions more efficiently, using a "Merkle tree"
  - Suppose basic hash from $\{0,1\}^k$ to $\{0,1\}^{k/2}$. A hash function from $\{0,1\}^{4k}$ to $\{0,1\}^{k/2}$ using a tree of depth 3
    - If basic hash from $\{0,1\}^k$ to $\{0,1\}^{k-1}$, first construct new basic hash from $\{0,1\}^k$ to $\{0,1\}^{k/2}$, by repeated hashing



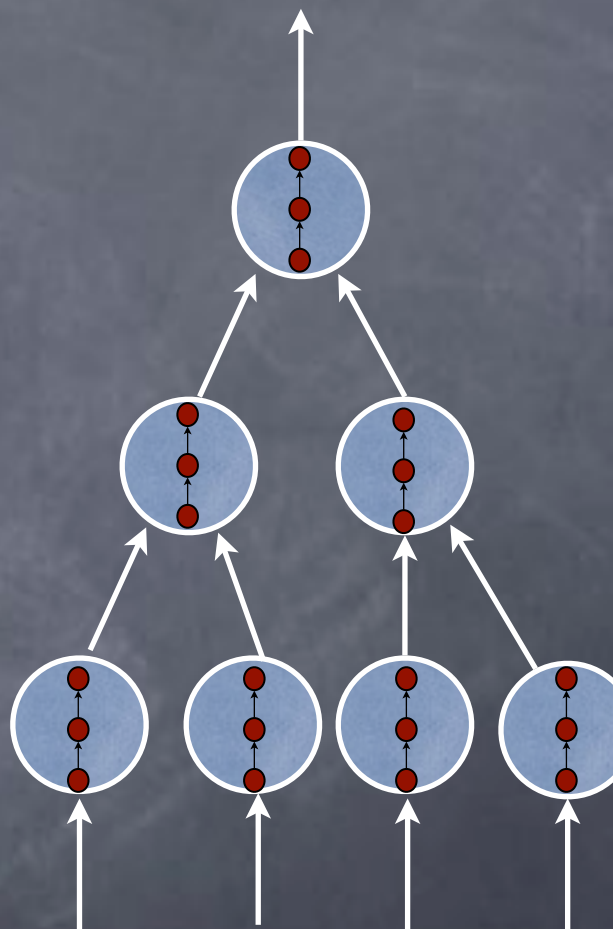
Domain Extension

- Can compose hash functions more efficiently, using a "Merkle tree"
  - Suppose basic hash from $\{0,1\}^k$ to $\{0,1\}^{k/2}$. A hash function from $\{0,1\}^{4k}$ to $\{0,1\}^{k/2}$ using a tree of depth 3
    - If basic hash from $\{0,1\}^k$ to $\{0,1\}^{k-1}$, first construct new basic hash from $\{0,1\}^k$ to $\{0,1\}^{k/2}$, by repeated hashing



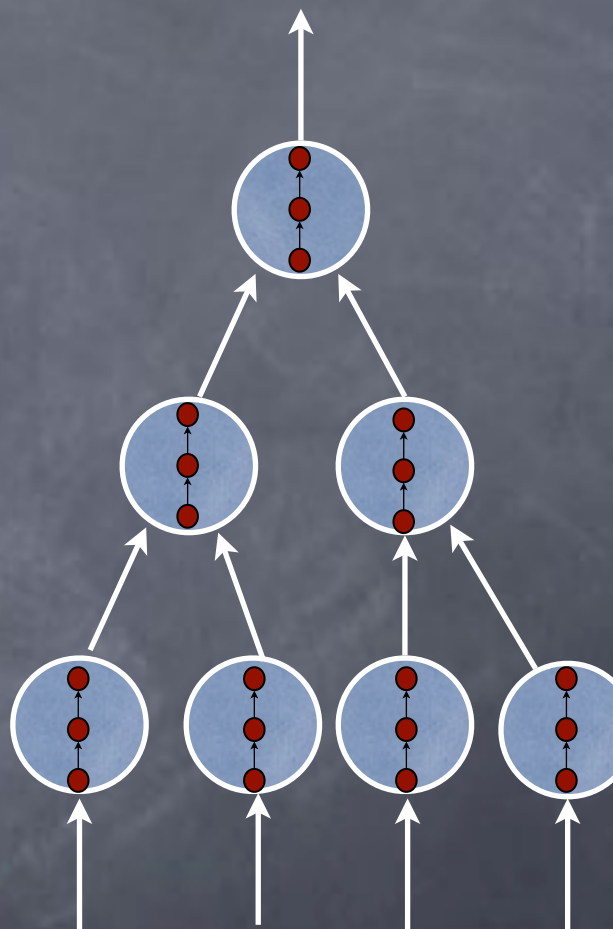
Domain Extension

- Can compose hash functions more efficiently, using a "Merkle tree"
  - Suppose basic hash from $\{0,1\}^k$ to $\{0,1\}^{k/2}$. A hash function from $\{0,1\}^{4k}$ to $\{0,1\}^{k/2}$ using a tree of depth 3
    - If basic hash from $\{0,1\}^k$ to $\{0,1\}^{k-1}$, first construct new basic hash from $\{0,1\}^k$ to $\{0,1\}^{k/2}$, by repeated hashing
  - Any tree can be used, with consistent I/O sizes



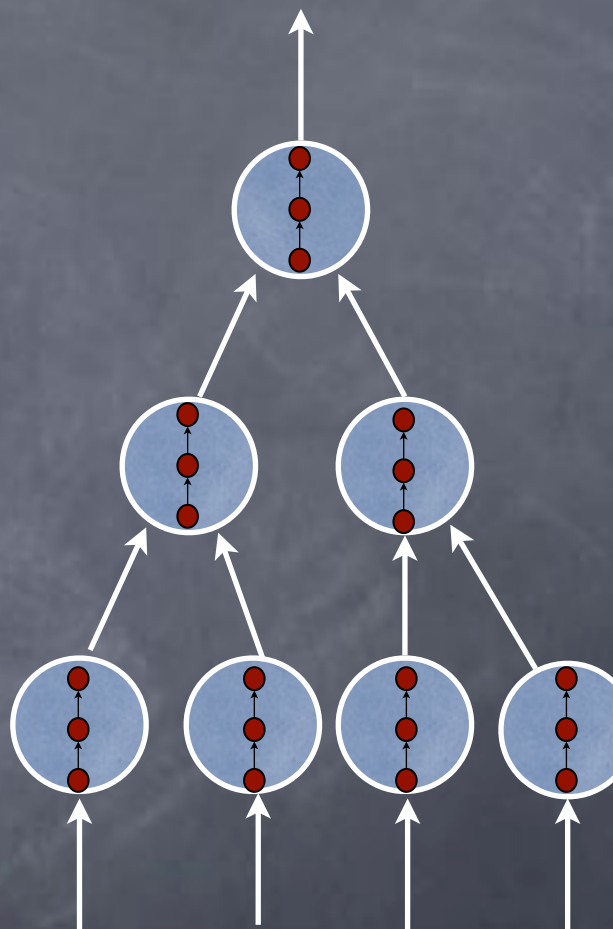
Domain Extension

- Can compose hash functions more efficiently, using a "Merkle tree"
  - Suppose basic hash from $\{0,1\}^k$ to $\{0,1\}^{k/2}$. A hash function from $\{0,1\}^{4k}$ to $\{0,1\}^{k/2}$ using a tree of depth 3
    - If basic hash from $\{0,1\}^k$ to $\{0,1\}^{k-1}$, first construct new basic hash from $\{0,1\}^k$ to $\{0,1\}^{k/2}$, by repeated hashing
  - Any tree can be used, with consistent I/O sizes
- Independent hashes or same hash?

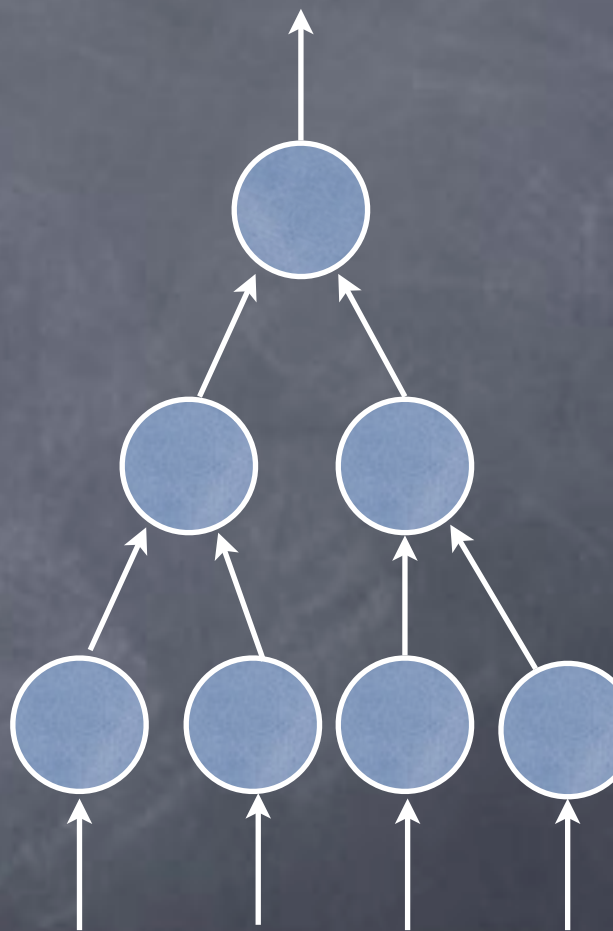


Domain Extension

- Can compose hash functions more efficiently, using a "Merkle tree"
  - Suppose basic hash from $\{0,1\}^k$ to $\{0,1\}^{k/2}$. A hash function from $\{0,1\}^{4k}$ to $\{0,1\}^{k/2}$ using a tree of depth 3
    - If basic hash from $\{0,1\}^k$ to $\{0,1\}^{k-1}$, first construct new basic hash from $\{0,1\}^k$ to $\{0,1\}^{k/2}$, by repeated hashing
  - Any tree can be used, with consistent I/O sizes
- Independent hashes or same hash?
  - Depends!

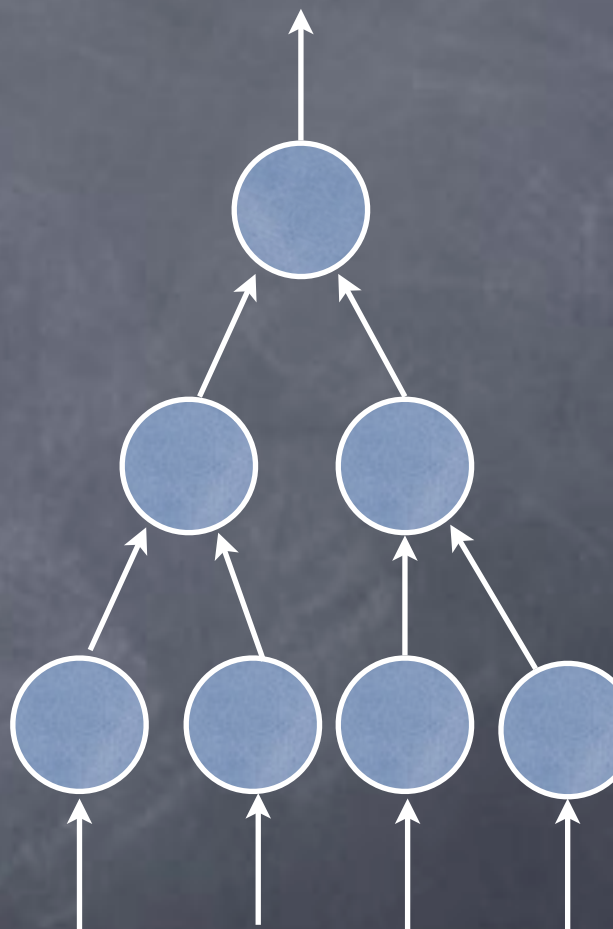


Domain Extension for CRHF



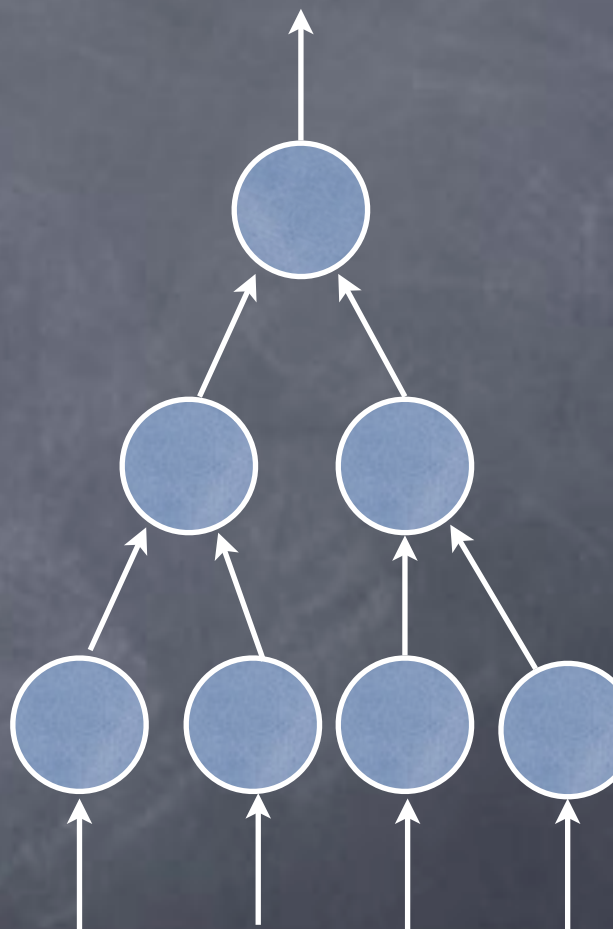
Domain Extension for CRHF

- For CRHF, same basic hash used through out the Merkle tree. Hash description same as for a single basic hash



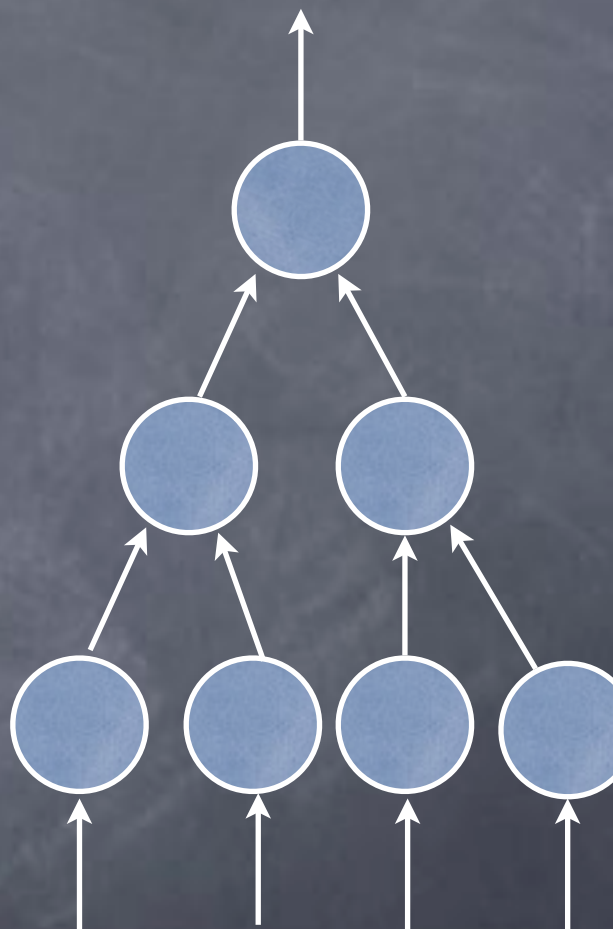
Domain Extension for CRHF

- For CRHF, same basic hash used through out the Merkle tree. Hash description same as for a single basic hash
- If a collision ($(x_1 \ldots x_n), (y_1 \ldots y_n)$) over all, then some collision (x', y') for basic hash



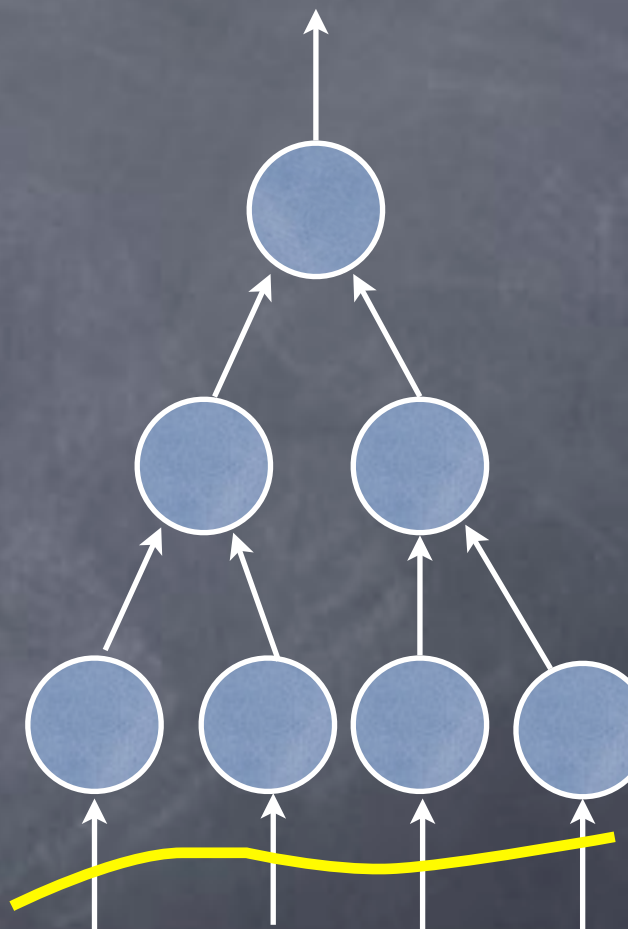
Domain Extension for CRHF

- For CRHF, same basic hash used through out the Merkle tree. Hash description same as for a single basic hash
- If a collision ($(x_1 \ldots x_n)$, $(y_1 \ldots y_n)$) over all, then some collision (x', y') for basic hash
  - Consider moving a "frontline" from bottom to top



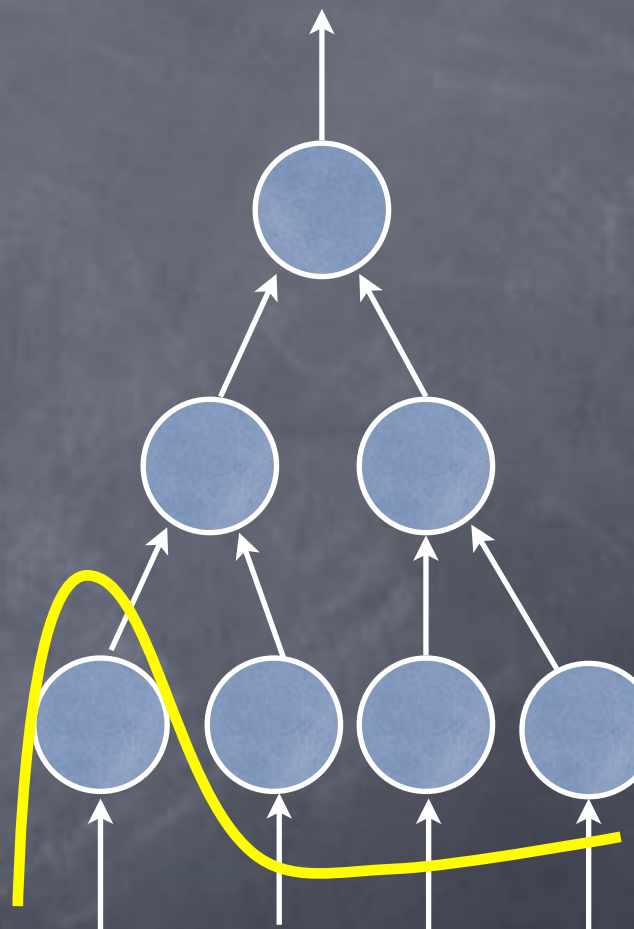
Domain Extension for CRHF

- For CRHF, same basic hash used through out the Merkle tree. Hash description same as for a single basic hash
- If a collision ($(x_1 \ldots x_n)$, $(y_1 \ldots y_n)$) over all, then some collision (x', y') for basic hash
  - Consider moving a "frontline" from bottom to top



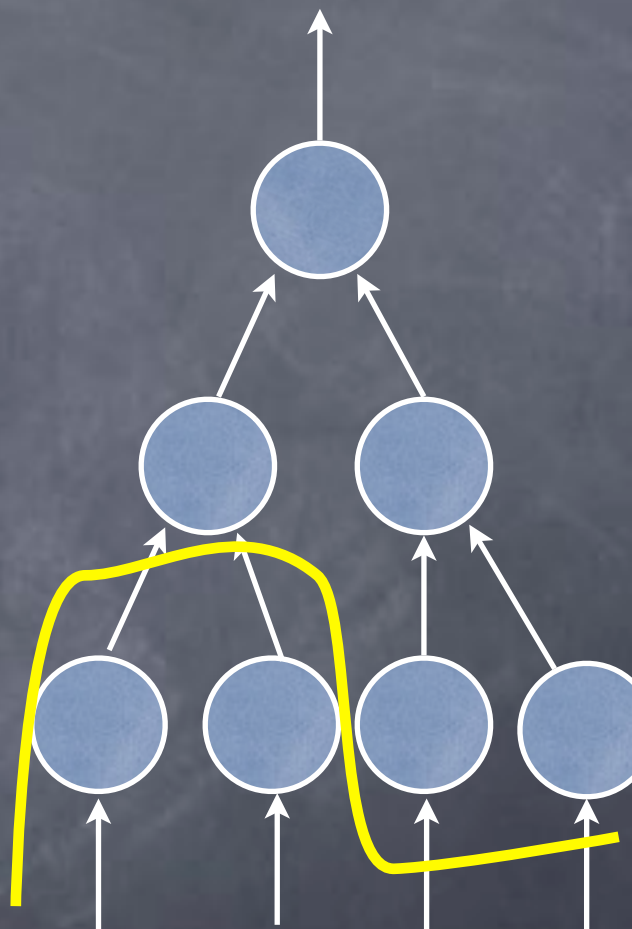
Domain Extension for CRHF

- For CRHF, same basic hash used through out the Merkle tree. Hash description same as for a single basic hash
- If a collision ($(x_1 \ldots x_n)$, $(y_1 \ldots y_n)$) over all, then some collision (x', y') for basic hash
  - Consider moving a "frontline" from bottom to top



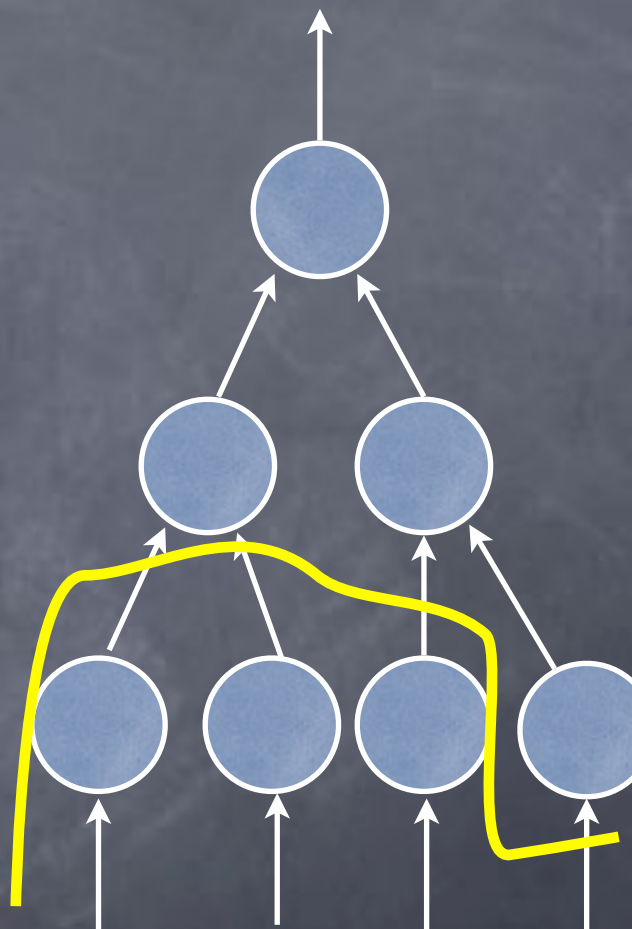
Domain Extension for CRHF

- For CRHF, same basic hash used through out the Merkle tree. Hash description same as for a single basic hash
- If a collision ($(x_1 \ldots x_n)$, $(y_1 \ldots y_n)$) over all, then some collision (x', y') for basic hash
  - Consider moving a "frontline" from bottom to top



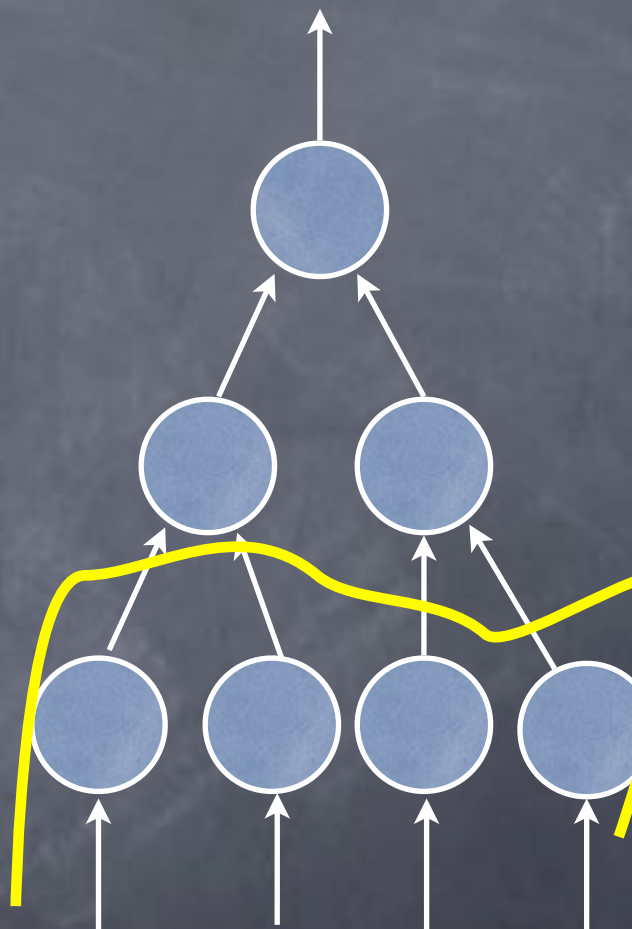
Domain Extension for CRHF

- For CRHF, same basic hash used through out the Merkle tree. Hash description same as for a single basic hash
- If a collision ($(x_1 \ldots x_n)$, $(y_1 \ldots y_n)$) over all, then some collision (x', y') for basic hash
  - Consider moving a "frontline" from bottom to top



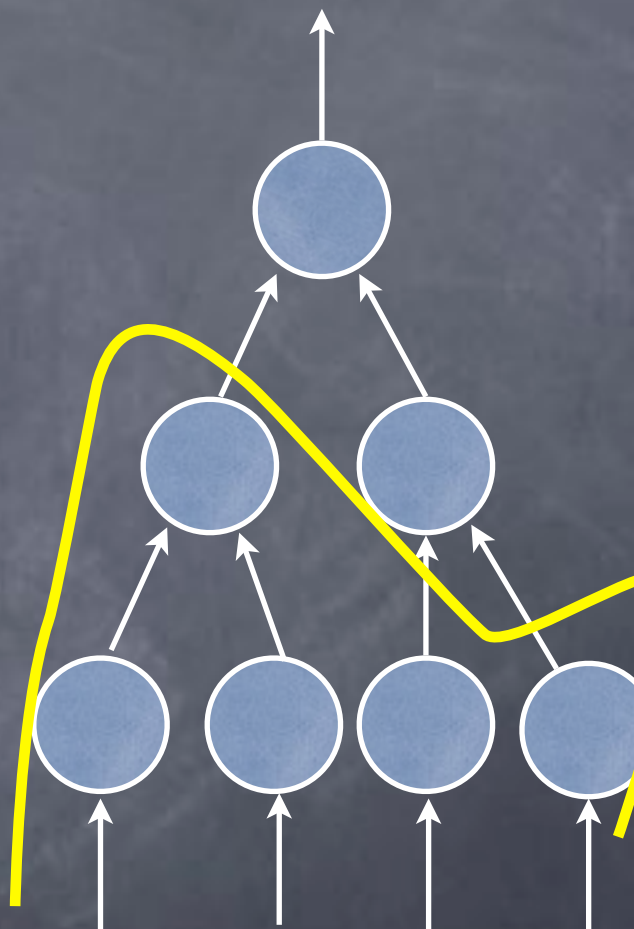
Domain Extension for CRHF

- For CRHF, same basic hash used through out the Merkle tree. Hash description same as for a single basic hash
- If a collision ($(x_1 \ldots x_n)$, $(y_1 \ldots y_n)$) over all, then some collision (x', y') for basic hash
  - Consider moving a "frontline" from bottom to top



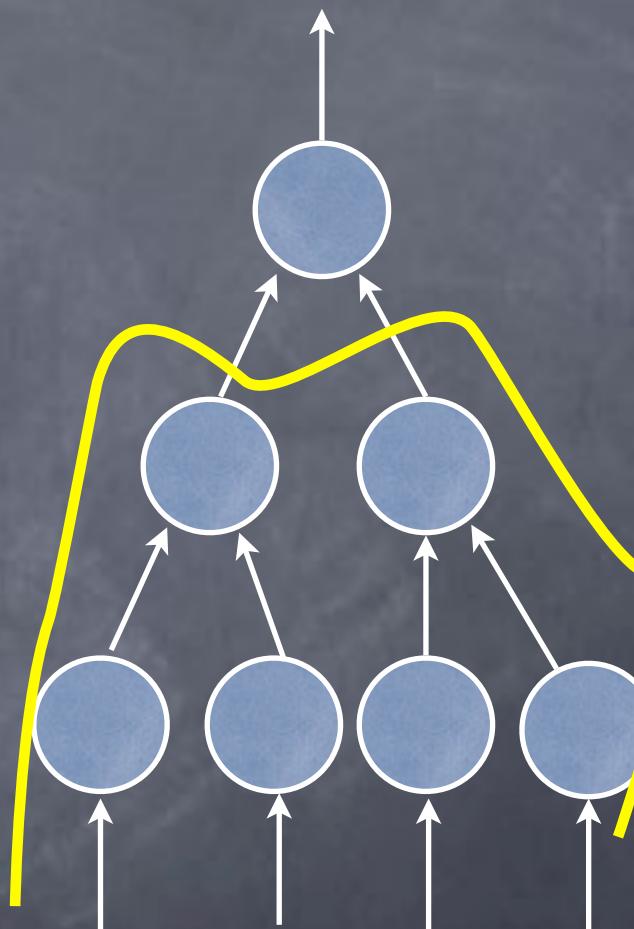
Domain Extension for CRHF

- For CRHF, same basic hash used through out the Merkle tree. Hash description same as for a single basic hash
- If a collision ($(x_1 \ldots x_n)$, $(y_1 \ldots y_n)$) over all, then some collision (x', y') for basic hash
  - Consider moving a "frontline" from bottom to top



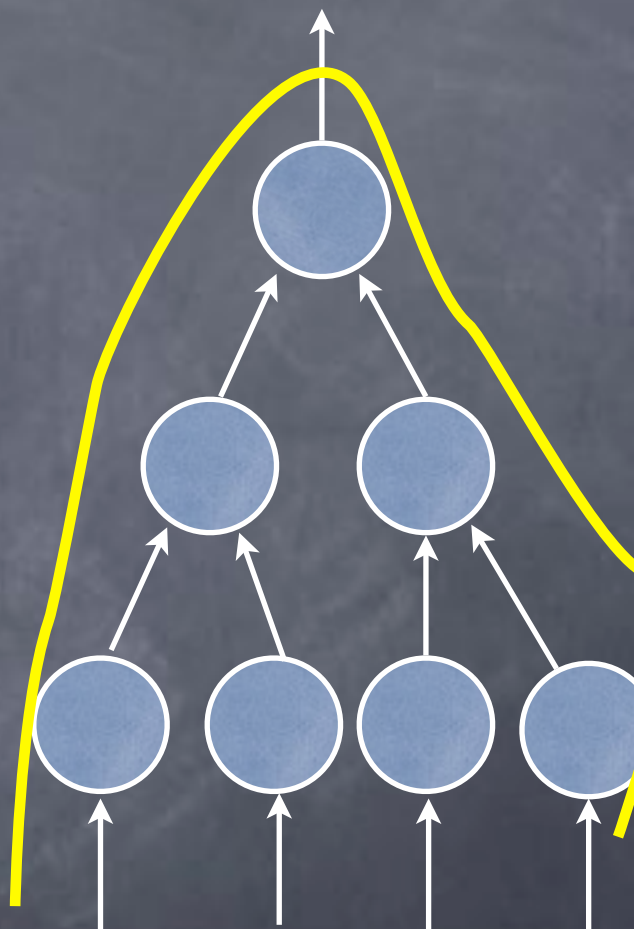
Domain Extension for CRHF

- For CRHF, same basic hash used through out the Merkle tree. Hash description same as for a single basic hash
- If a collision ($(x_1 \ldots x_n)$, $(y_1 \ldots y_n)$) over all, then some collision (x', y') for basic hash
  - Consider moving a "frontline" from bottom to top



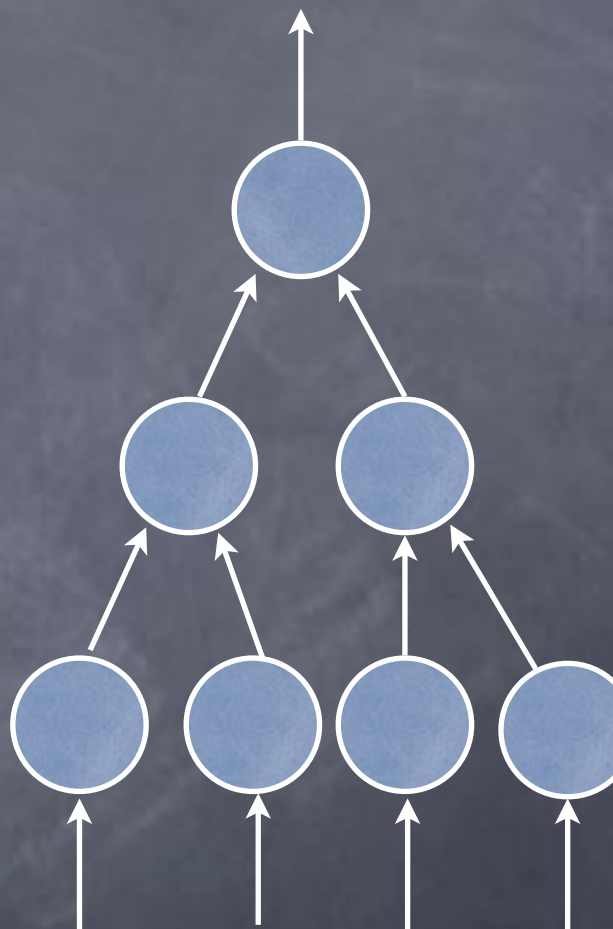
Domain Extension for CRHF

- For CRHF, same basic hash used through out the Merkle tree. Hash description same as for a single basic hash
- If a collision ($(x_1 \ldots x_n)$, $(y_1 \ldots y_n)$) over all, then some collision (x', y') for basic hash
  - Consider moving a "frontline" from bottom to top



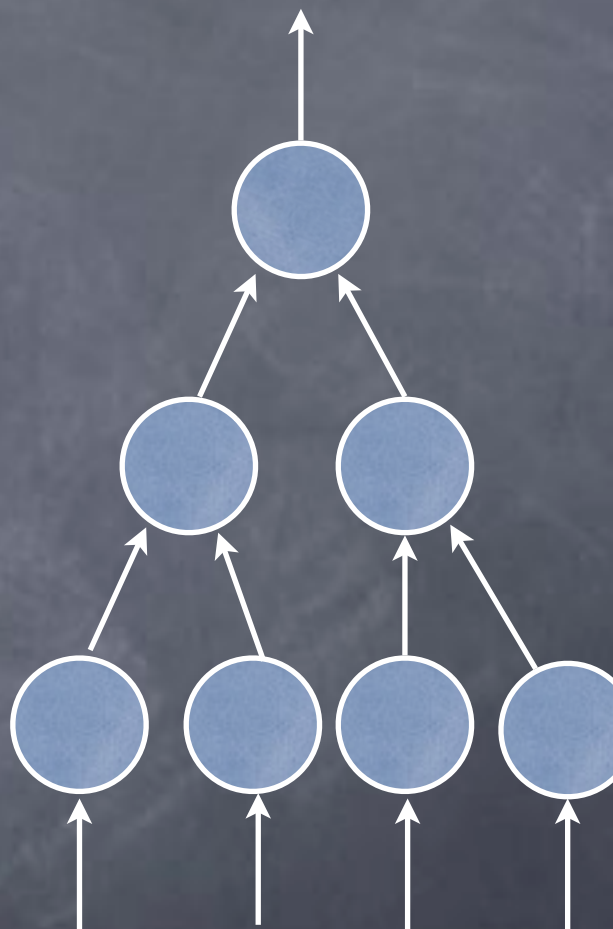
Domain Extension for CRHF

- For CRHF, same basic hash used through out the Merkle tree. Hash description same as for a single basic hash
- If a collision ($(x_1 \ldots x_n)$, $(y_1 \ldots y_n)$) over all, then some collision (x', y') for basic hash
  - Consider moving a "frontline" from bottom to top



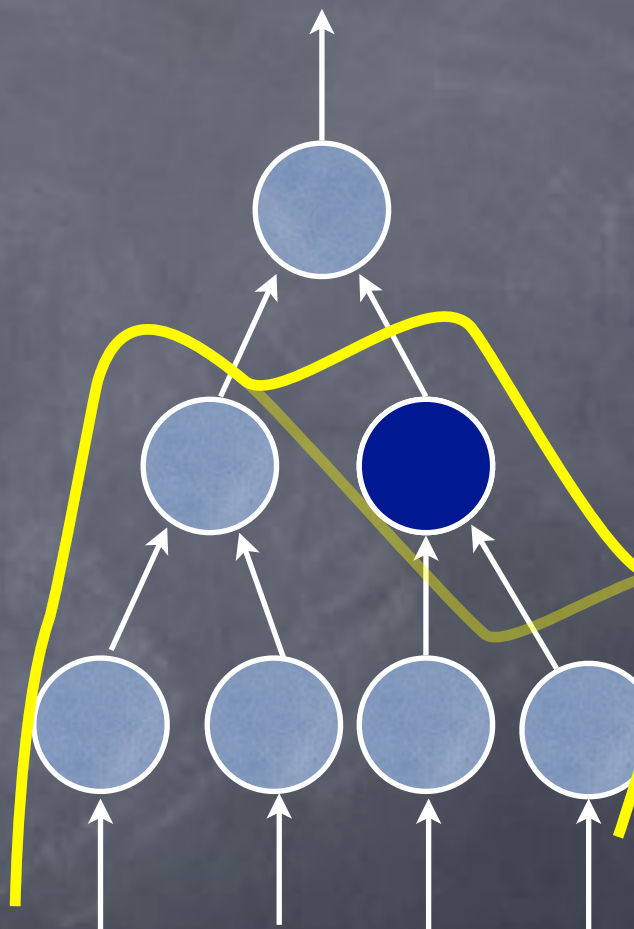
Domain Extension for CRHF

- For CRHF, same basic hash used through out the Merkle tree. Hash description same as for a single basic hash
- If a collision ($(x_1 \ldots x_n)$, $(y_1 \ldots y_n)$) over all, then some collision (x', y') for basic hash
  - Consider moving a "frontline" from bottom to top
    - Collision at some step (different values on i^{th} front, same on $i+1^{st}$); gives a collision for basic hash



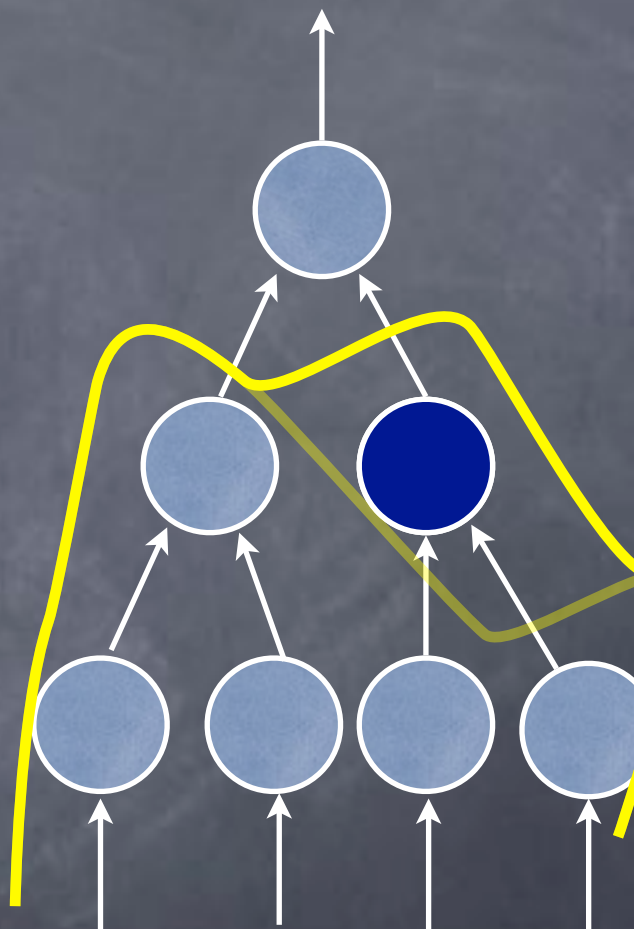
Domain Extension for CRHF

- For CRHF, same basic hash used through out the Merkle tree. Hash description same as for a single basic hash
- If a collision ($(x_1 \ldots x_n)$, $(y_1 \ldots y_n)$) over all, then some collision (x', y') for basic hash
  - Consider moving a "frontline" from bottom to top
    - Collision at some step (different values on i^{th} front, same on $i+1^{st}$); gives a collision for basic hash

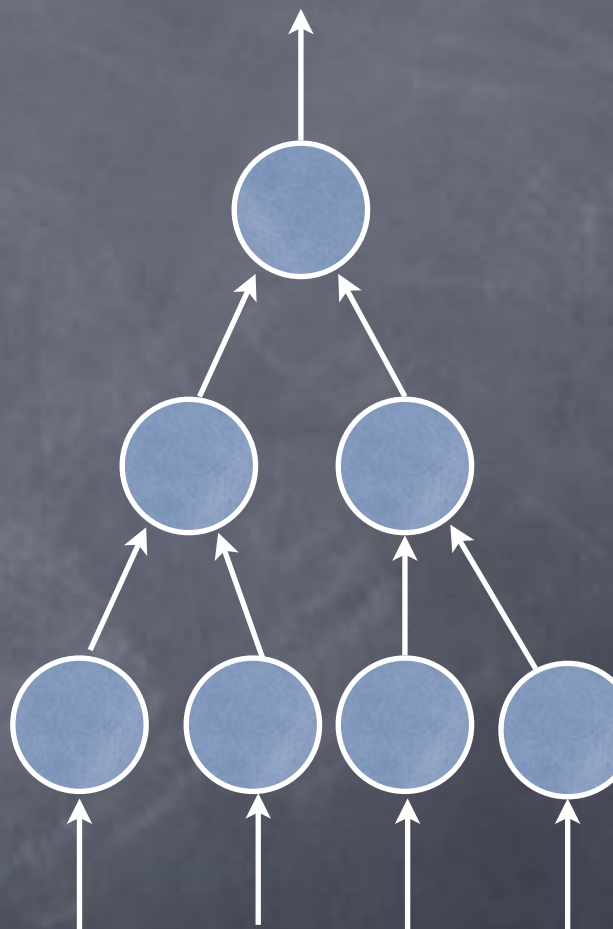


Domain Extension for CRHF

- For CRHF, same basic hash used through out the Merkle tree. Hash description same as for a single basic hash
- If a collision ($(x_1 \ldots x_n)$, $(y_1 \ldots y_n)$) over all, then some collision (x', y') for basic hash
  - Consider moving a "frontline" from bottom to top
    - Collision at some step (different values on i^{th} front, same on $i+1^{st}$); gives a collision for basic hash
- A*(h): run A(h) to get $(x_1 \ldots x_n)$, $(y_1 \ldots y_n)$. Move frontline to find (x', y')

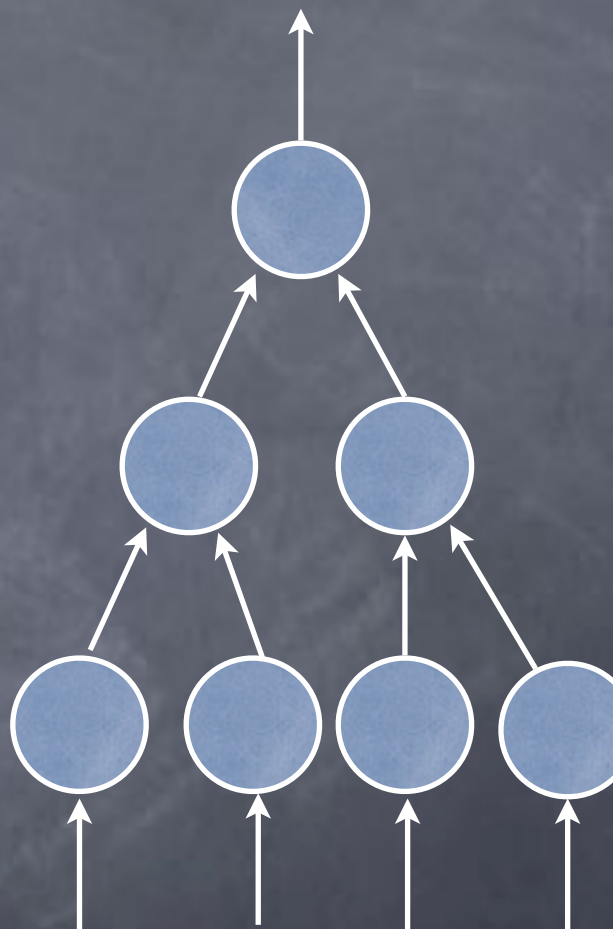


Domain Extension for UOWHF



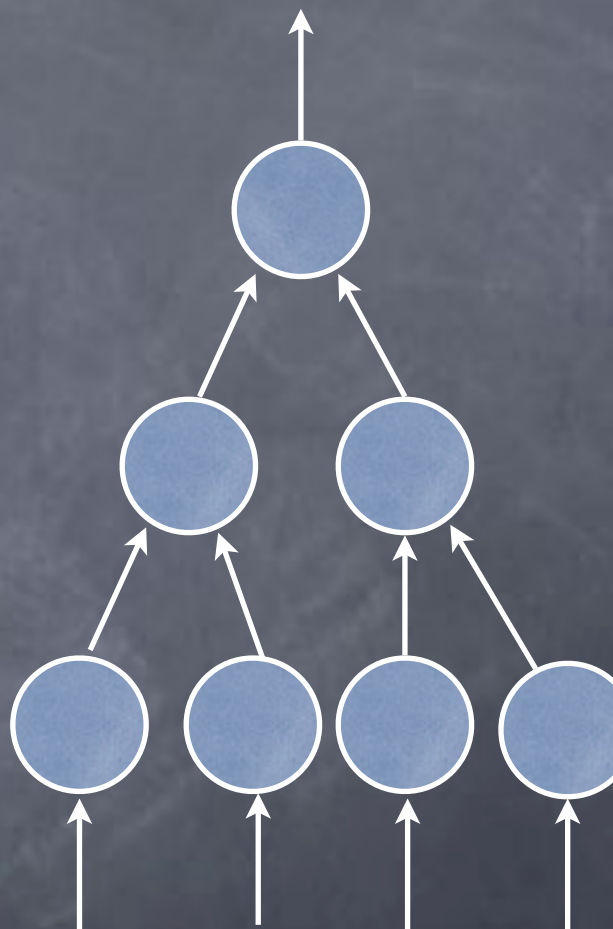
Domain Extension for UOWHF

- For UOWHF, can't use same basic hash throughout!



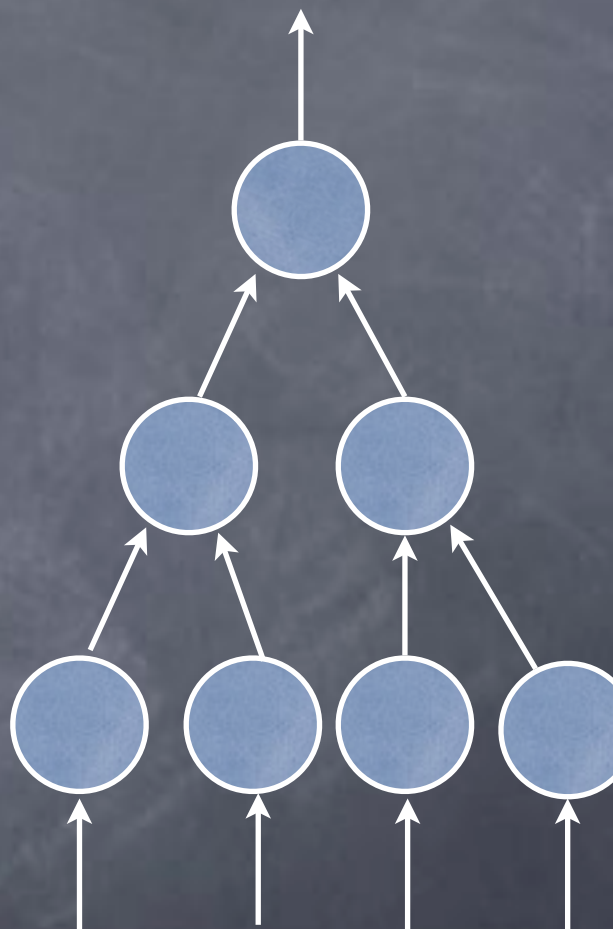
Domain Extension for UOWHF

- For UOWHF, can't use same basic hash throughout!
- A* has to output an x′ on getting $(x_1 \ldots x_n)$ from A, before getting h



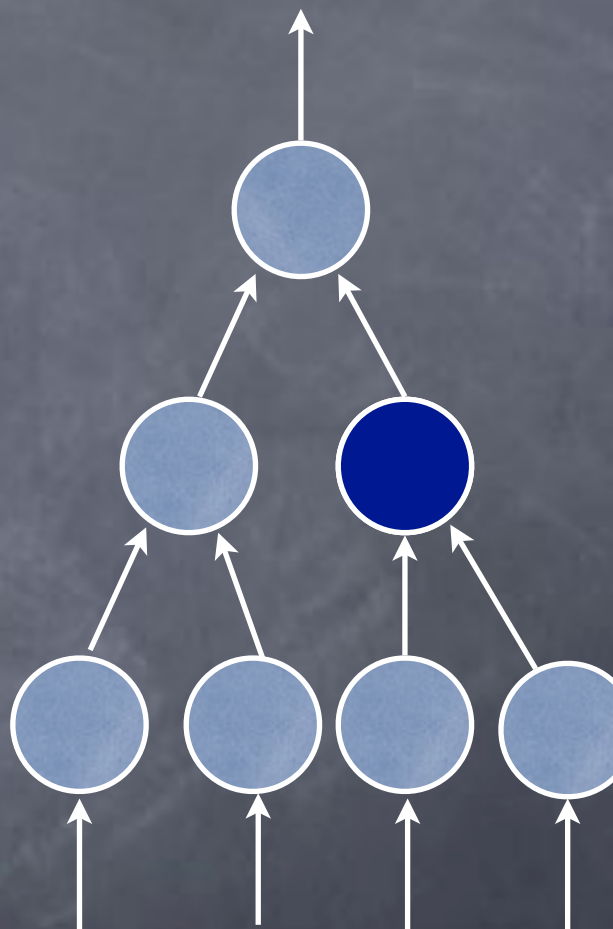
Domain Extension for UOWHF

- For UOWHF, can't use same basic hash throughout!
- A* has to output an x′ on getting $(x_1 \ldots x_n)$ from A, before getting h
  - Can guess a random node (i.e., random pair of frontlines) where collision occurs, but can't compute x′ until h is fixed!



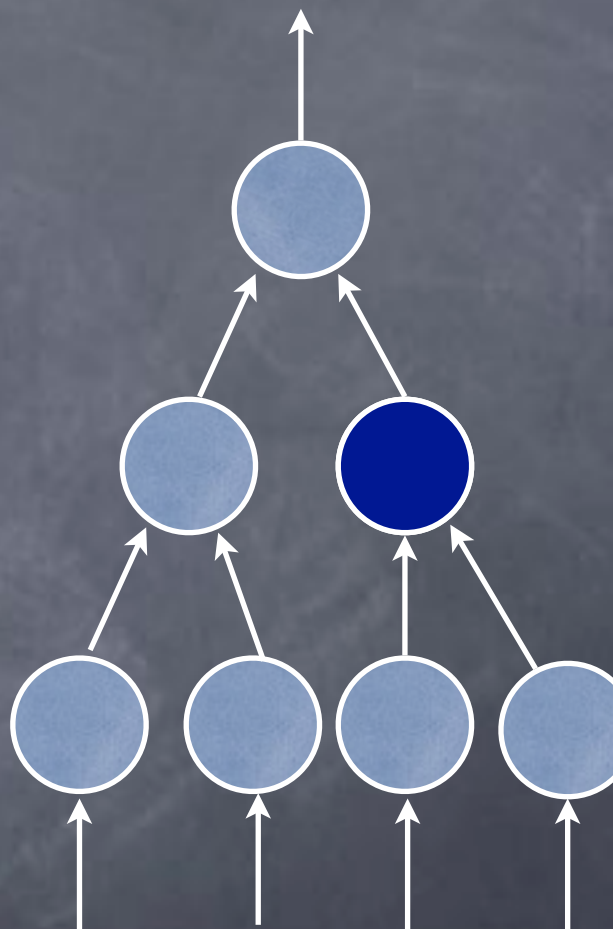
Domain Extension for UOWHF

- For UOWHF, can't use same basic hash throughout!
- A* has to output an x′ on getting $(x_1 \ldots x_n)$ from A, before getting h
  - Can guess a random node (i.e., random pair of frontlines) where collision occurs, but can't compute x′ until h is fixed!



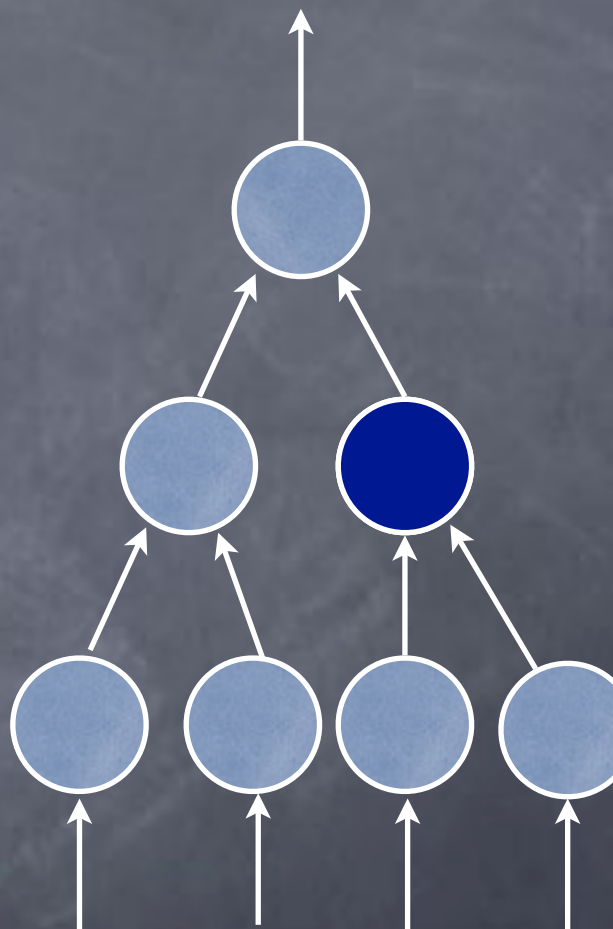
Domain Extension for UOWHF

- For UOWHF, can't use same basic hash throughout!
- A* has to output an x′ on getting $(x_1 \ldots x_n)$ from A, before getting h
  - Can guess a random node (i.e., random pair of frontlines) where collision occurs, but can't compute x′ until h is fixed!
- Solution: a different h for each level of the tree (i.e., no ancestor/successor has same h)



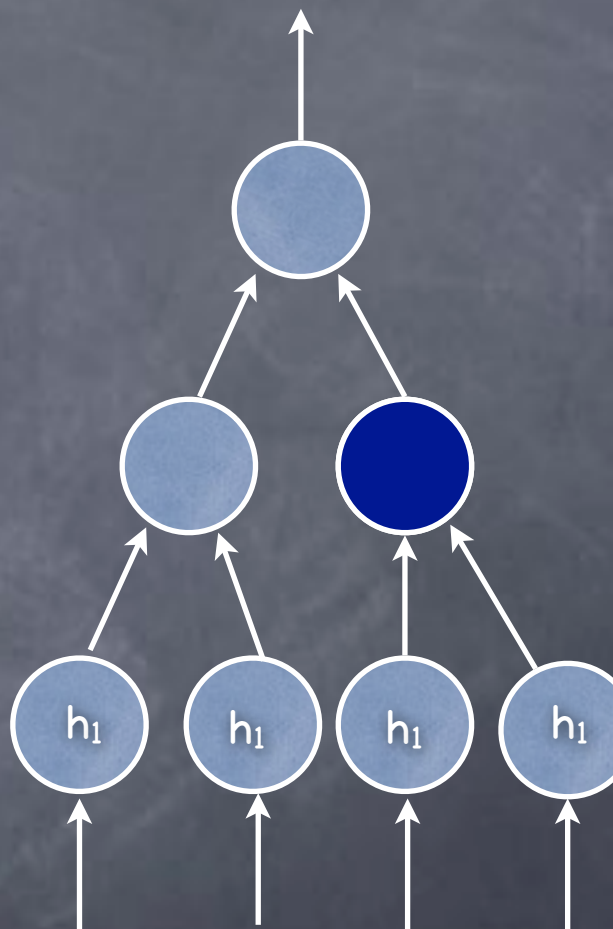
Domain Extension for UOWHF

- For UOWHF, can't use same basic hash throughout!
- A* has to output an x′ on getting $(x_1 \ldots x_n)$ from A, before getting h
  - Can guess a random node (i.e., random pair of frontlines) where collision occurs, but can't compute x′ until h is fixed!
- Solution: a different h for each level of the tree (i.e., no ancestor/successor has same h)
  - To compute x′: pick a random node (say at level i) pick h_j for levels below i, and from $(x_1 \ldots x_n)$ compute input to the node; this be x′



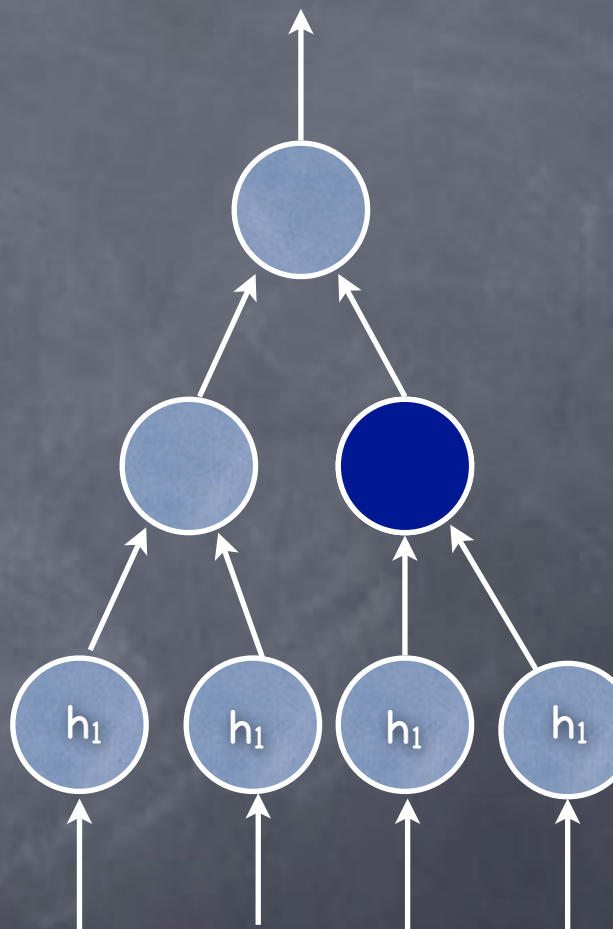
Domain Extension for UOWHF

- For UOWHF, can't use same basic hash throughout!
- A* has to output an x′ on getting $(x_1 \ldots x_n)$ from A, before getting h
  - Can guess a random node (i.e., random pair of frontlines) where collision occurs, but can't compute x′ until h is fixed!
- Solution: a different h for each level of the tree (i.e., no ancestor/successor has same h)
  - To compute x′: pick a random node (say at level i) pick h_j for levels below i, and from $(x_1 \ldots x_n)$ compute input to the node; this be x′



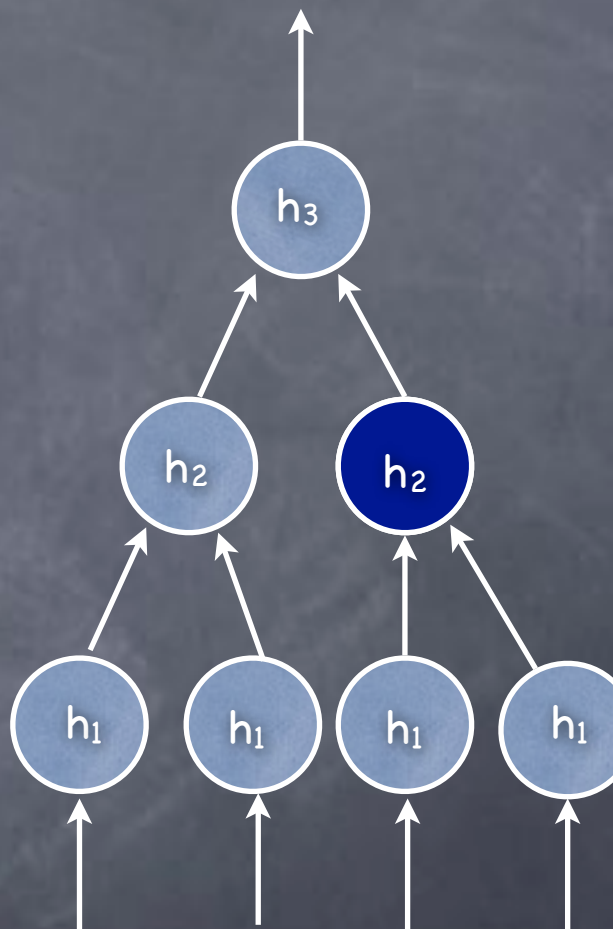
Domain Extension for UOWHF

- For UOWHF, can't use same basic hash throughout!
- A* has to output an x′ on getting $(x_1 \ldots x_n)$ from A, before getting h
  - Can guess a random node (i.e., random pair of frontlines) where collision occurs, but can't compute x′ until h is fixed!
- Solution: a different h for each level of the tree (i.e., no ancestor/successor has same h)
  - To compute x′: pick a random node (say at level i) pick h_j for levels below i, and from $(x_1 \ldots x_n)$ compute input to the node; this be x′
  - On getting h, plug it in as h_i, pick h_j for remaining levels; get $(y_1 \ldots y_n)$ and compute y′



Domain Extension for UOWHF

- For UOWHF, can't use same basic hash throughout!
- A* has to output an x′ on getting $(x_1 \ldots x_n)$ from A, before getting h
  - Can guess a random node (i.e., random pair of frontlines) where collision occurs, but can't compute x′ until h is fixed!
- Solution: a different h for each level of the tree (i.e., no ancestor/successor has same h)
  - To compute x′: pick a random node (say at level i) pick h_j for levels below i, and from $(x_1 \ldots x_n)$ compute input to the node; this be x′
  - On getting h, plug it in as h_i, pick h_j for remaining levels; get $(y_1 \ldots y_n)$ and compute y′



UOWHF vs. CRHF

UOWHF vs. CRHF

- UOWHF has a weaker guarantee than CRHF

UOWHF vs. CRHF

- UOWHF has a weaker guarantee than CRHF
- UOWHF can be built based on OWF (we saw based on OWP), where as CRHF "needs stronger assumptions"

UOWHF vs. CRHF

- UOWHF has a weaker guarantee than CRHF
- UOWHF can be built based on OWF (we saw based on OWP), where as CRHF "needs stronger assumptions"
  - But "usual" OWF candidates suffice for CRHF too (we saw construction based on discrete-log)

UOWHF vs. CRHF

- UOWHF has a weaker guarantee than CRHF
- UOWHF can be built based on OWF (we saw based on OWP), where as CRHF "needs stronger assumptions"
  - But "usual" OWF candidates suffice for CRHF too (we saw construction based on discrete-log)
- Domain extension of CRHF is simpler, with no blow-up in the description size. For UOWHF description increases logarithmically in the input size

UOWHF vs. CRHF

- UOWHF has a weaker guarantee than CRHF
- UOWHF can be built based on OWF (we saw based on OWP), where as CRHF "needs stronger assumptions"
  - But "usual" OWF candidates suffice for CRHF too (we saw construction based on discrete-log)
- Domain extension of CRHF is simpler, with no blow-up in the description size. For UOWHF description increases logarithmically in the input size
- UOWHF theoretically important (based on simpler assumptions, good if paranoid), but CRHF can substitute for it

UOWHF vs. CRHF

- UOWHF has a weaker guarantee than CRHF
- UOWHF can be built based on OWF (we saw based on OWP), where as CRHF "needs stronger assumptions"
  - But "usual" OWF candidates suffice for CRHF too (we saw construction based on discrete-log)
- Domain extension of CRHF is simpler, with no blow-up in the description size. For UOWHF description increases logarithmically in the input size
- UOWHF theoretically important (based on simpler assumptions, good if paranoid), but CRHF can substitute for it
- Current practice: much less paranoid; faith on efficient, ad hoc (and unkeyed) constructions (though increasingly under attack)

Today

Today

- Combinatorial hash functions, UOWHF and CRHF

Today

- Combinatorial hash functions, UOWHF and CRHF
  - (And weaker variants of CRHF: pre-image collision resistance and second-pre-image collision resistance)

Today

- Combinatorial hash functions, UOWHF and CRHF
  - (And weaker variants of CRHF: pre-image collision resistance and second-pre-image collision resistance)
- Collision-resistant combinatorial HF from 2-Universal Hash Functions

Today

- Combinatorial hash functions, UOWHF and CRHF
  - (And weaker variants of CRHF: pre-image collision resistance and second-pre-image collision resistance)
- Collision-resistant combinatorial HF from 2-Universal Hash Functions
- UOWHF from UHF and OWP (possible from OWF)

Today

- Combinatorial hash functions, UOWHF and CRHF
  - (And weaker variants of CRHF: pre-image collision resistance and second-pre-image collision resistance)
- Collision-resistant combinatorial HF from 2-Universal Hash Functions
- UOWHF from UHF and OWP (possible from OWF)
- CRHF from discrete-log (possible from several hardness assumptions; stronger than OWP)

Today

- Combinatorial hash functions, UOWHF and CRHF
  - (And weaker variants of CRHF: pre-image collision resistance and second-pre-image collision resistance)
- Collision-resistant combinatorial HF from 2-Universal Hash Functions
- UOWHF from UHF and OWP (possible from OWF)
- CRHF from discrete-log (possible from several hardness assumptions; stronger than OWP)
- Domain extension for CRHF and UOWHF, using Merkle-Trees

Today

- Combinatorial hash functions, UOWHF and CRHF
  - (And weaker variants of CRHF: pre-image collision resistance and second-pre-image collision resistance)
- Collision-resistant combinatorial HF from 2-Universal Hash Functions
- UOWHF from UHF and OWP (possible from OWF)
- CRHF from discrete-log (possible from several hardness assumptions; stronger than OWP)
- Domain extension for CRHF and UOWHF, using Merkle-Trees
- Next lecture: Using hash functions