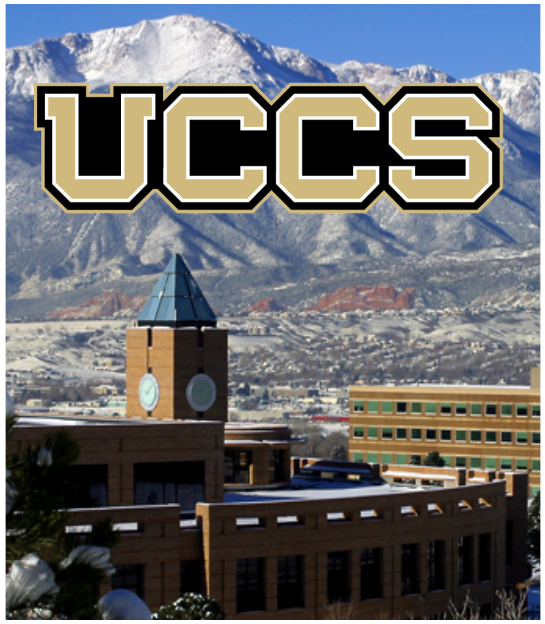


Cryptographic Hash and Integrity Protection

Cryptographic Hash Applications

Sang-Yoon Chang, Ph.D.

Module: Cryptographic Hash Applications

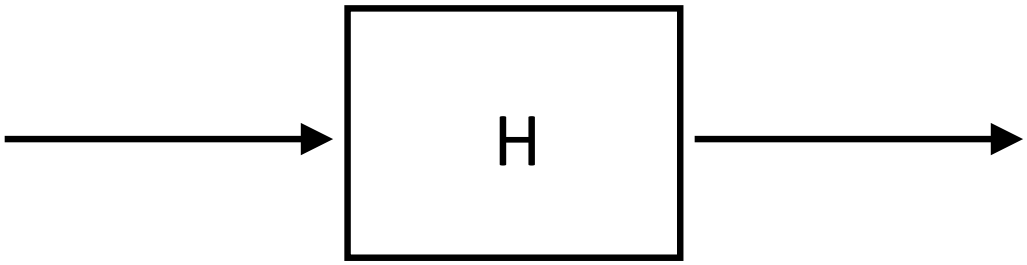
Hash Chain

S/Key: One-Time Password

Hash Tree

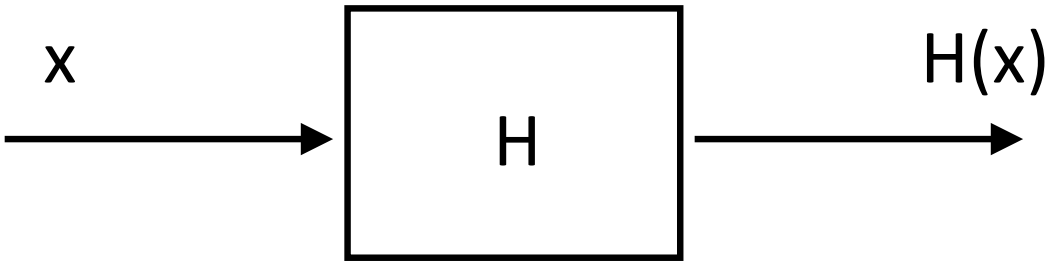
Cryptocurrency and Bitcoin

Cryptographic Hash Applications



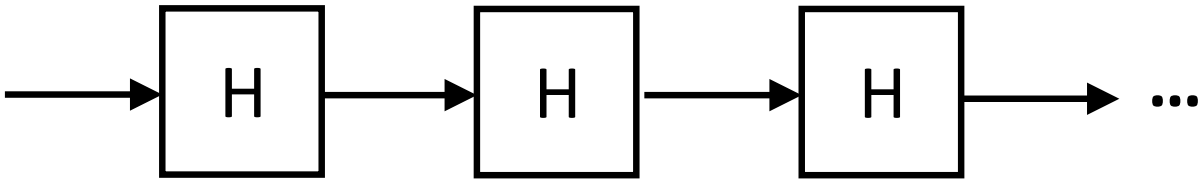
Hash Function

Cryptographic Hash Applications

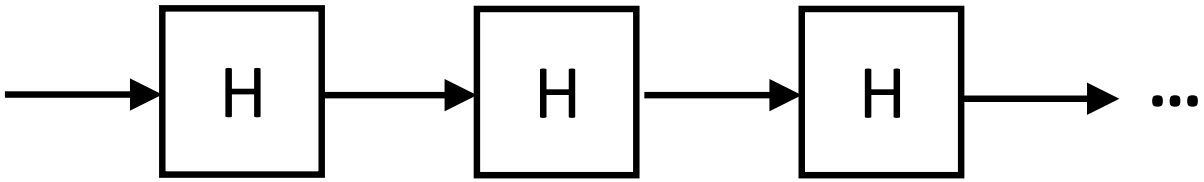


Hash Function

Hash Chain



Hash Chain

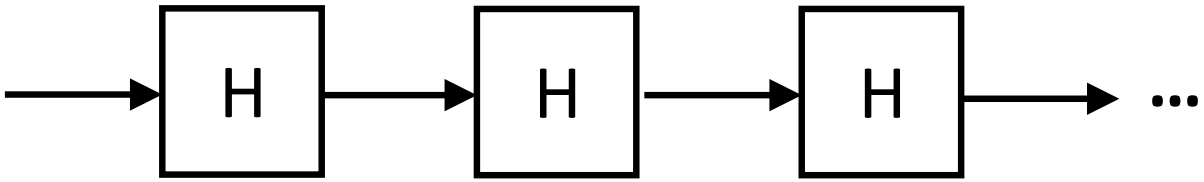


$H^n(x)$: apply H function on x n times

E.g., $H^3(x) = H(H(H(x)))$

One-way (preimage resistance) of H

Hash Chain

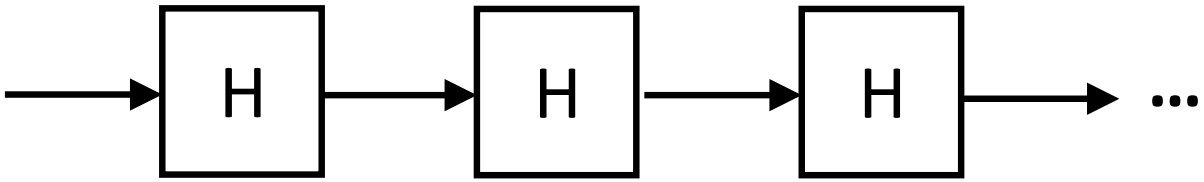


$H^n(x)$: apply H function on x n times

E.g., $H^3(x) = H(H(H(x)))$

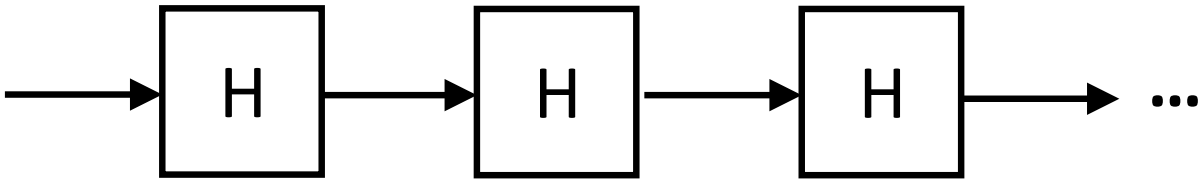
One-way (preimage resistance) for H

S/Key: One-Time Password Generation



Server (verifier) generates
$x, H(x), H^2(x), \ldots, H^{n+1}(x)$

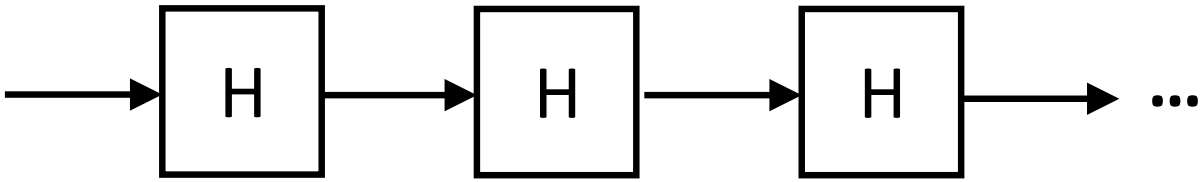
S/Key: One-Time Password Generation



Server (verifier) generates

~~x~~, $H(x)$, $H^2(x)$, …, $H^{n+1}(x)$

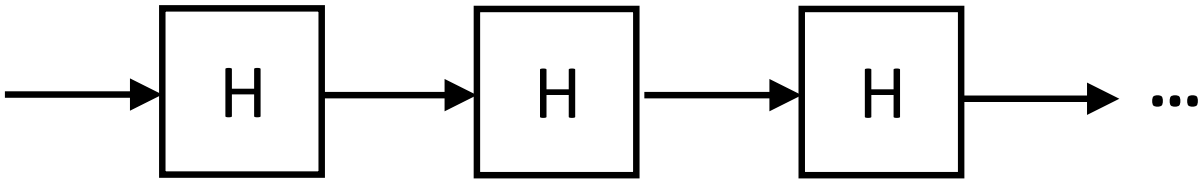
S/Key: One-Time Password Generation



Server (verifier) generates

$H(x), H^2(x), \ldots, H^{n+1}(x)$

S/Key: One-Time Password Generation



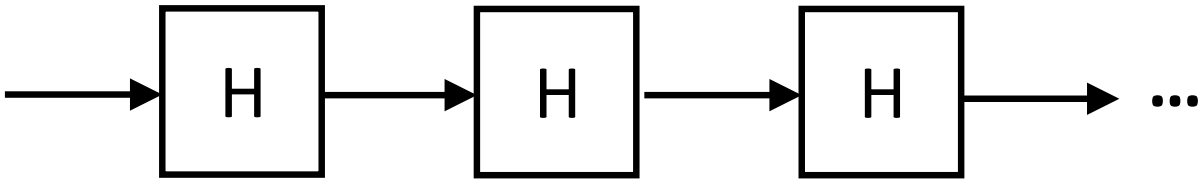
Server (verifier) generates

$H(x), H^2(x), \ldots, H^{n+1}(x)$

User (prover) is given n passwords:

$H(x), H^2(x), \ldots, H^n(x)$

S/Key: One-Time Password Generation



Server (verifier) generates

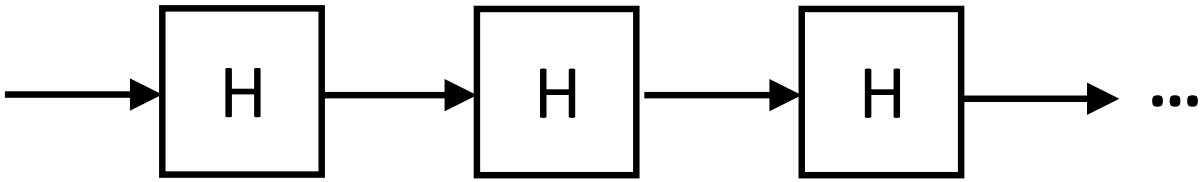
$H(x), H^2(x), \ldots, \underline{H^{n+1}(x)}$

Server only stores $H^{n+1}(x)$

User (prover) is given n passwords:

$H(x), H^2(x), \ldots, H^n(x)$

S/Key: One-Time Password Generation



Server (verifier) generates

$H(x), H^2(x), \ldots, \underline{H^{n+1}(x)}$

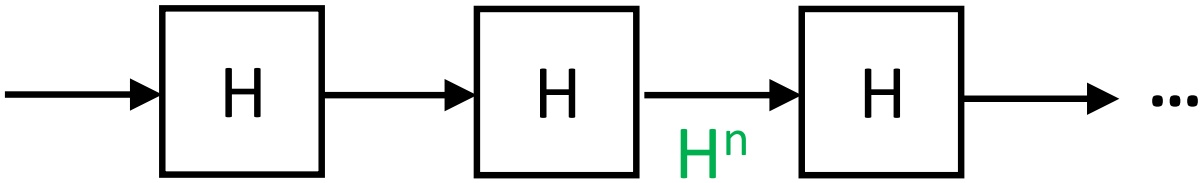
Server only stores $H^{n+1}(x)$

User (prover) is given n passwords:

$H(x), H^2(x), \ldots, H^n(x)$

Uses in the reverse order

S/Key: One-Time Password Authentication



Server (verifier) generates

$H(x), H^2(x), \ldots, H^{n+1}(x)$

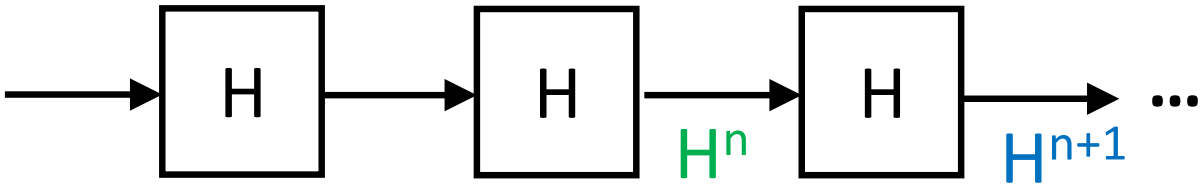
Server only stores $H^{n+1}(x)$

User (prover) is given n passwords:

$H(x), H^2(x), \ldots, H^n(x)$

Uses in the reverse order

S/Key: One-Time Password Authentication



Server (verifier) generates

$H(x), H^2(x), \ldots, \underline{H^{n+1}(x)}$

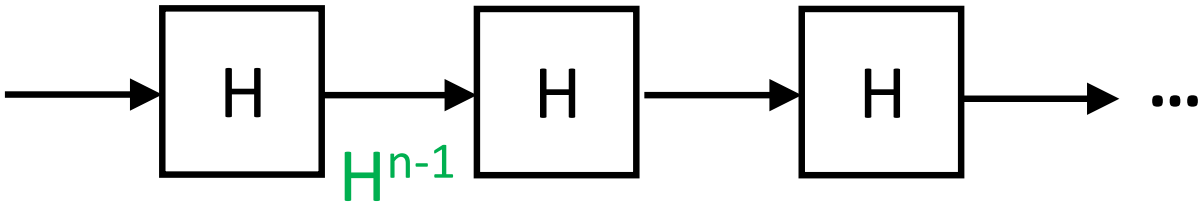
Server only stores $H^{n+1}(x)$

User (prover) is given n passwords:

$H(x), H^2(x), \ldots, H^n(x)$

Uses in the reverse order

S/Key: One-Time Password Authentication



Server (verifier) generates

$H(x), H^2(x), \ldots, \underline{H^{n+1}(x)}$

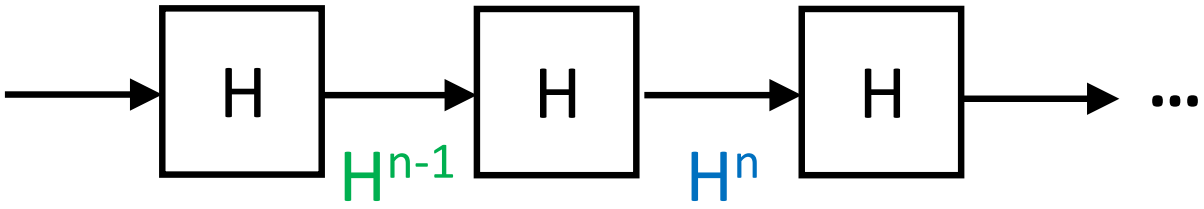
Server only stores $H^{n+1}(x)$

User (prover) is given n passwords:

$H(x), H^2(x), \ldots, H^n(x)$

Uses in the reverse order

S/Key: One-Time Password Authentication



Server (verifier) generates

$H(x), H^2(x), \ldots, \underline{H^{n+1}(x)}$

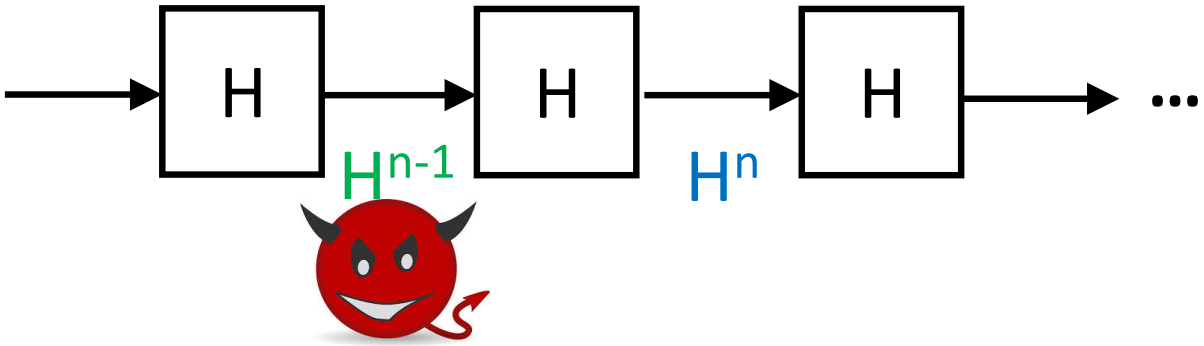
Server only stores $H^{n+1}(x)$

User (prover) is given n passwords:

$H(x), H^2(x), \ldots, H^n(x)$

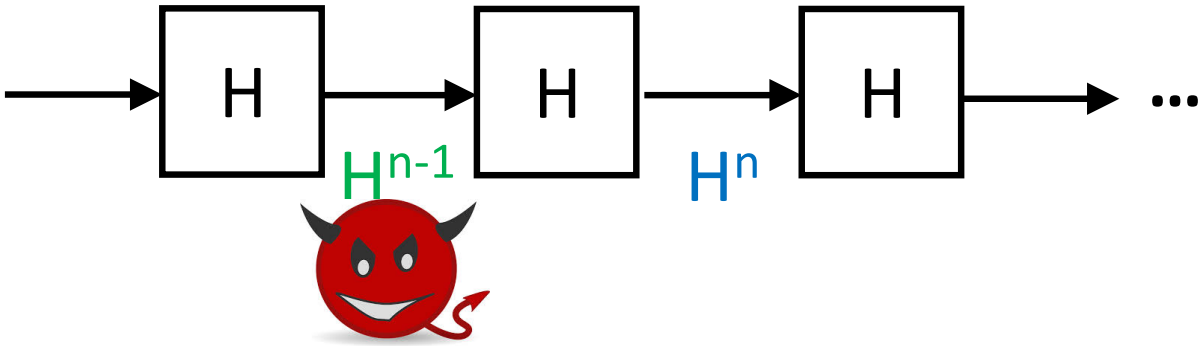
Uses in the reverse order

S/Key: One-Time Password Authentication



Compromise yields outdated passwords

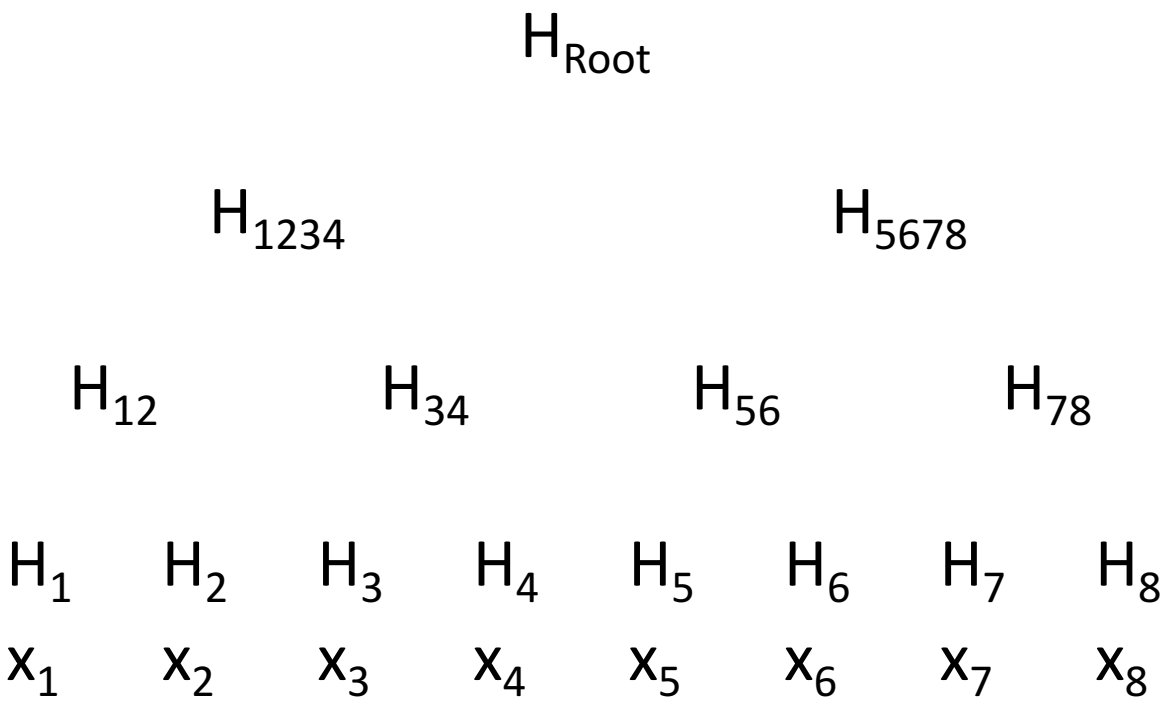
S/Key: One-Time Password Authentication



Compromise yields outdated passwords

Limits the number passwords to n

Hash Tree (Merkle Tree)



Hash Tree

$H(x_1)$ $H(x_2)$ $H(x_3)$ $H(x_4)$ $H(x_5)$ $H(x_6)$ $H(x_7)$ $H(x_8)$

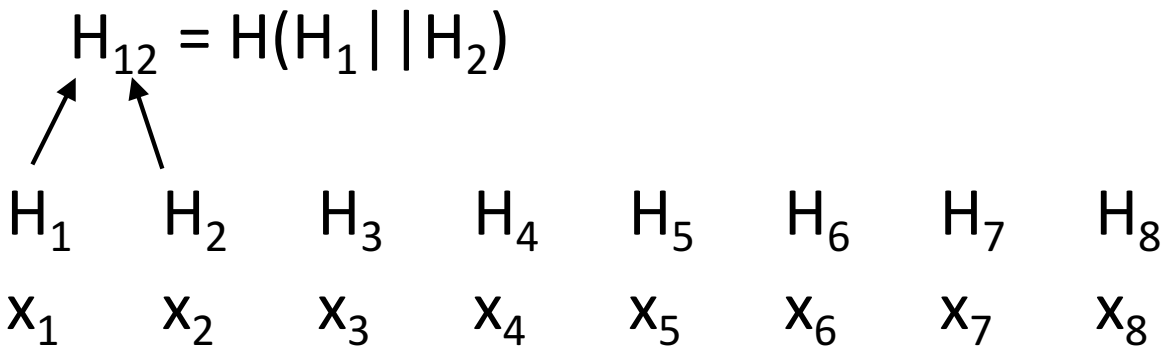
x_1 x_2 x_3 x_4 x_5 x_6 x_7 x_8

Hash Tree

H_1 H_2 H_3 H_4 H_5 H_6 H_7 H_8

x_1 x_2 x_3 x_4 x_5 x_6 x_7 x_8

Hash Tree



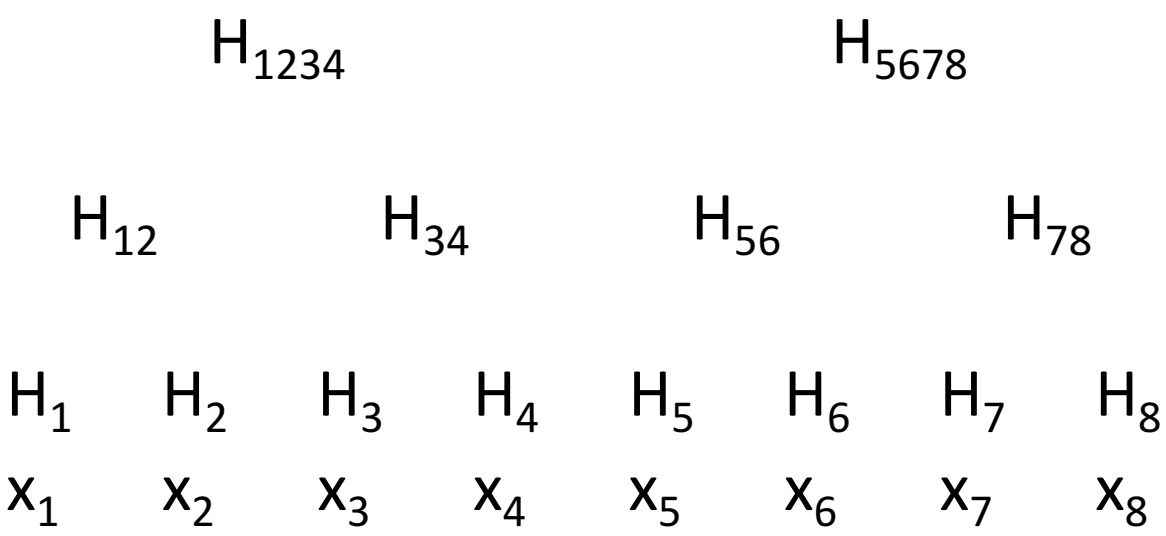
Hash Tree

H_{12} H_{34} H_{56} H_{78}

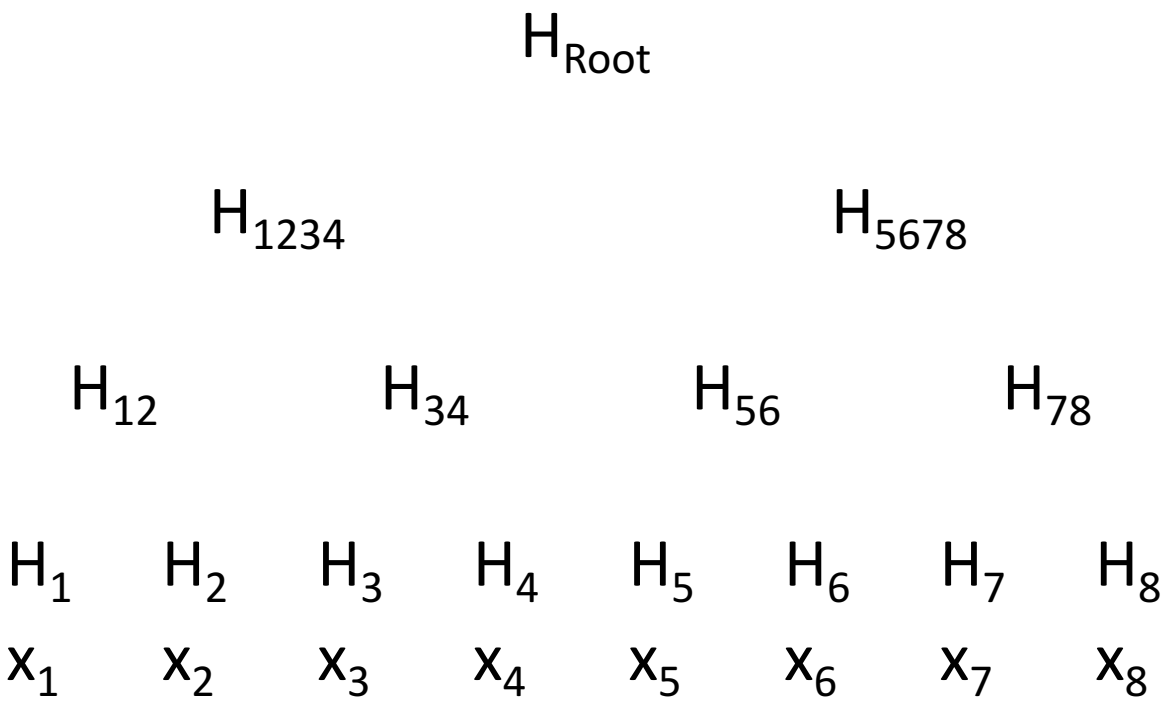
H_1 H_2 H_3 H_4 H_5 H_6 H_7 H_8

x_1 x_2 x_3 x_4 x_5 x_6 x_7 x_8

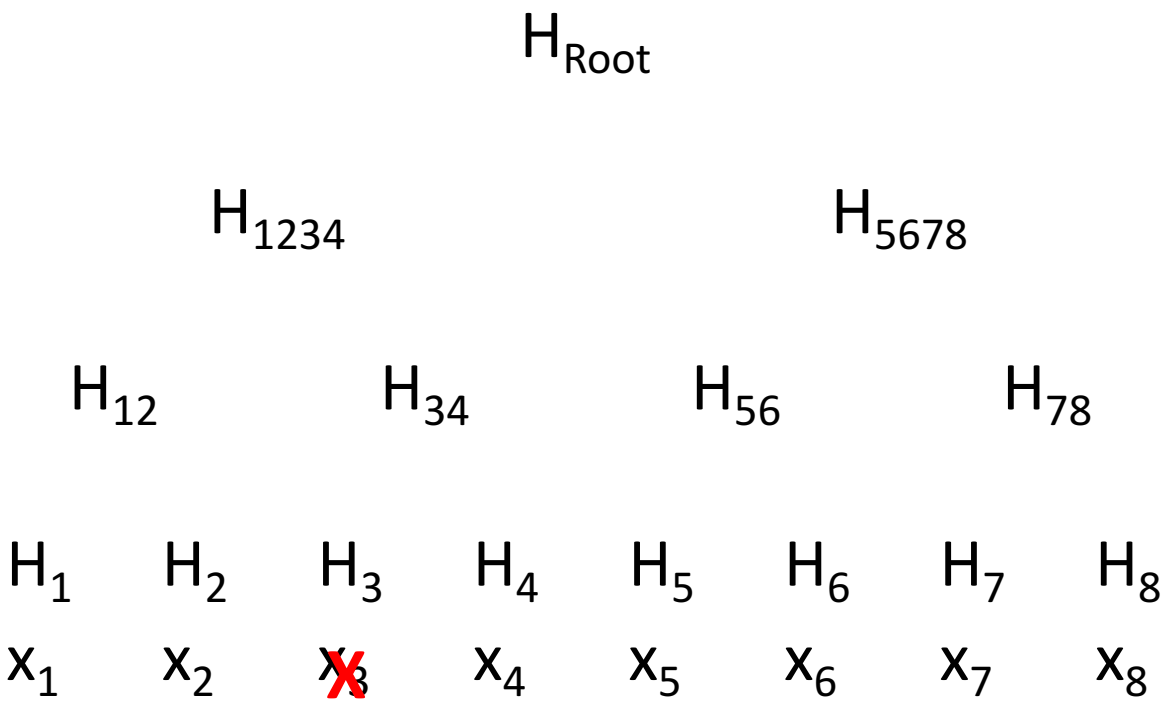
Hash Tree


H_{1234}
H_{5678}
H_{12}
H_{34}
H_{56}
H_{78}
H_1 H_2 H_3 H_4 H_5 H_6 H_7 H_8
x_1 x_2 x_3 x_4 x_5 x_6 x_7 x_8

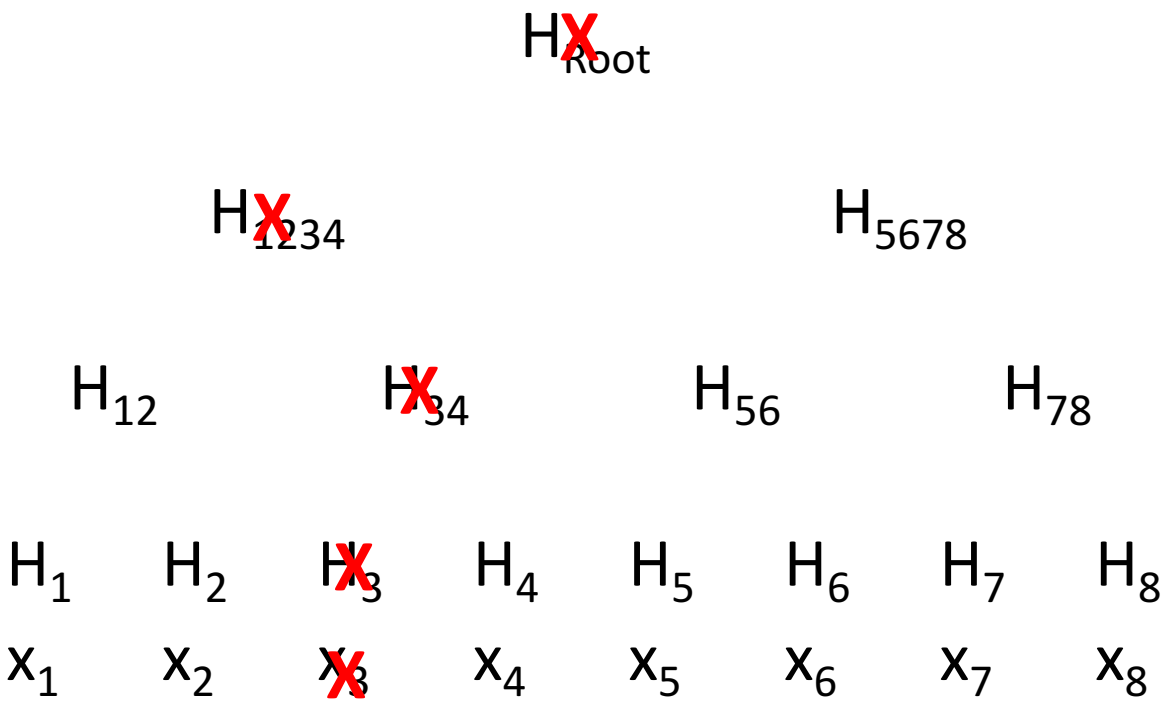
Hash Tree


H_{Root}
H_{1234}
H_{5678}
H_{12}
H_{34}
H_{56}
H_{78}
H_1 H_2 H_3 H_4 H_5 H_6 H_7 H_8
x_1 x_2 x_3 x_4 x_5 x_6 x_7 x_8

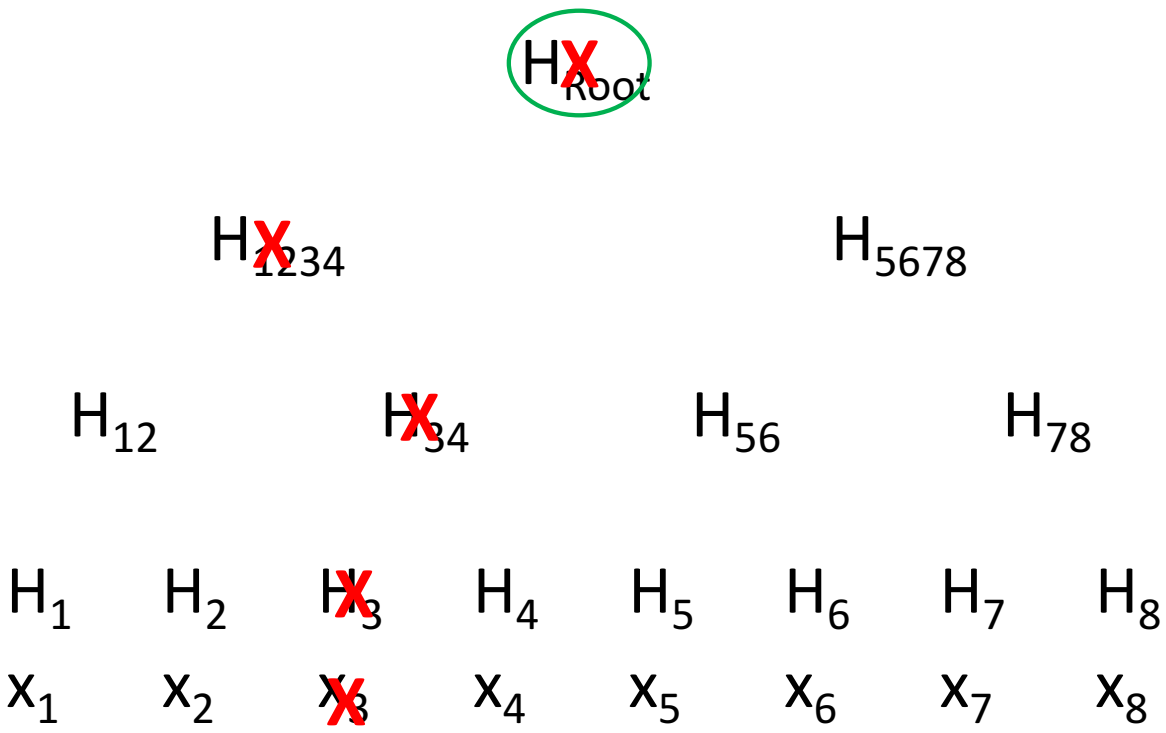
Hash Tree


H_{Root}
H_{1234}
H_{5678}
H_{12}
H_{34}
H_{56}
H_{78}
H_1 H_2 H_3 H_4 H_5 H_6 H_7 H_8
x_1 x_2 x_3 x_4 x_5 x_6 x_7 x_8

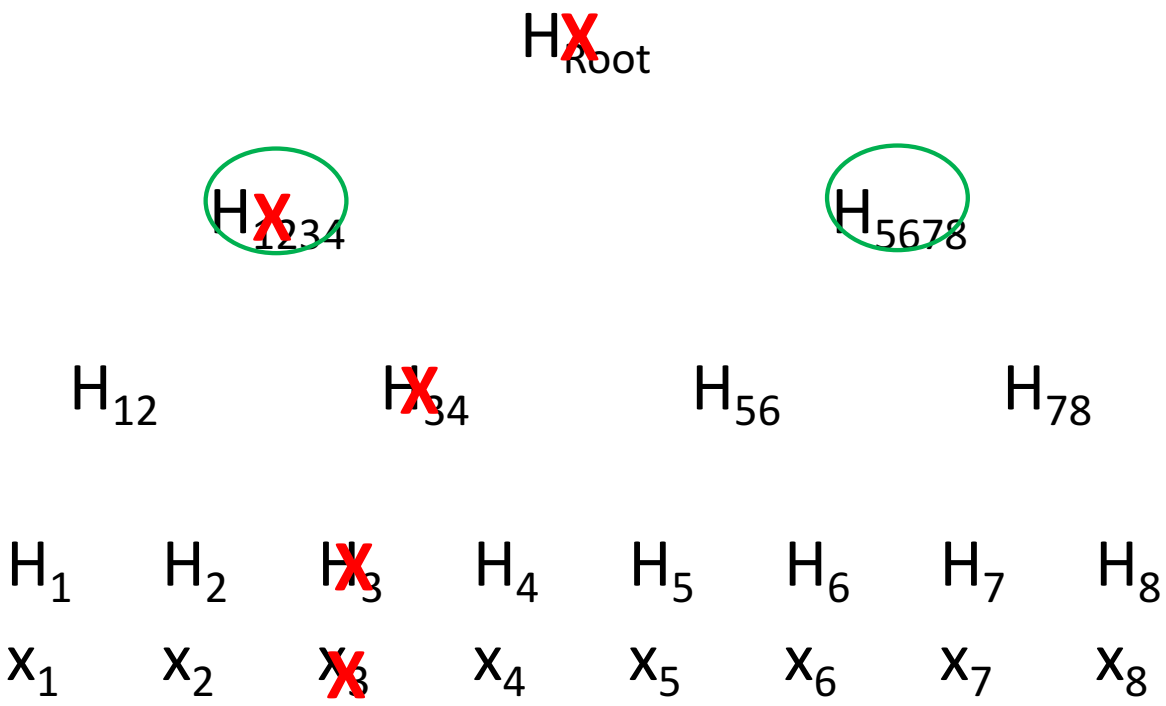
Hash Tree


H_{Root}
H_{1234}
H_{5678}
H_{12}
H_{34}
H_{56}
H_{78}
H_1 H_2 H_3 H_4 H_5 H_6 H_7 H_8
x_1 x_2 x_3 x_4 x_5 x_6 x_7 x_8

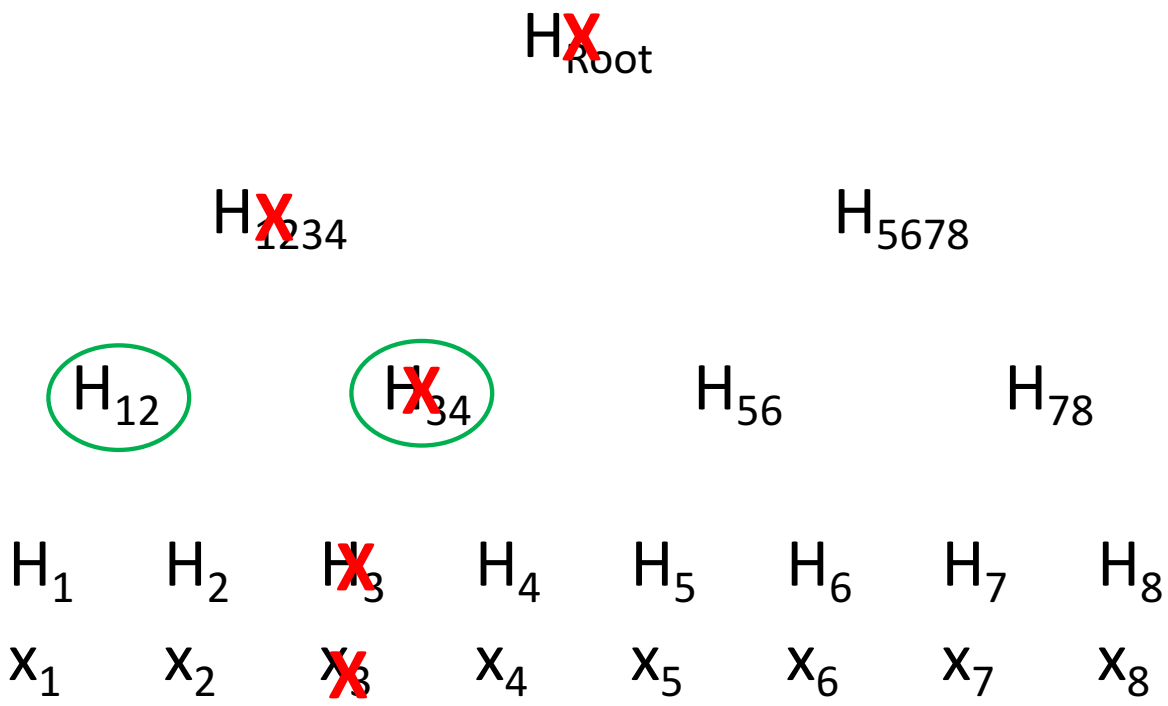
Hash Tree – Integrity Detection



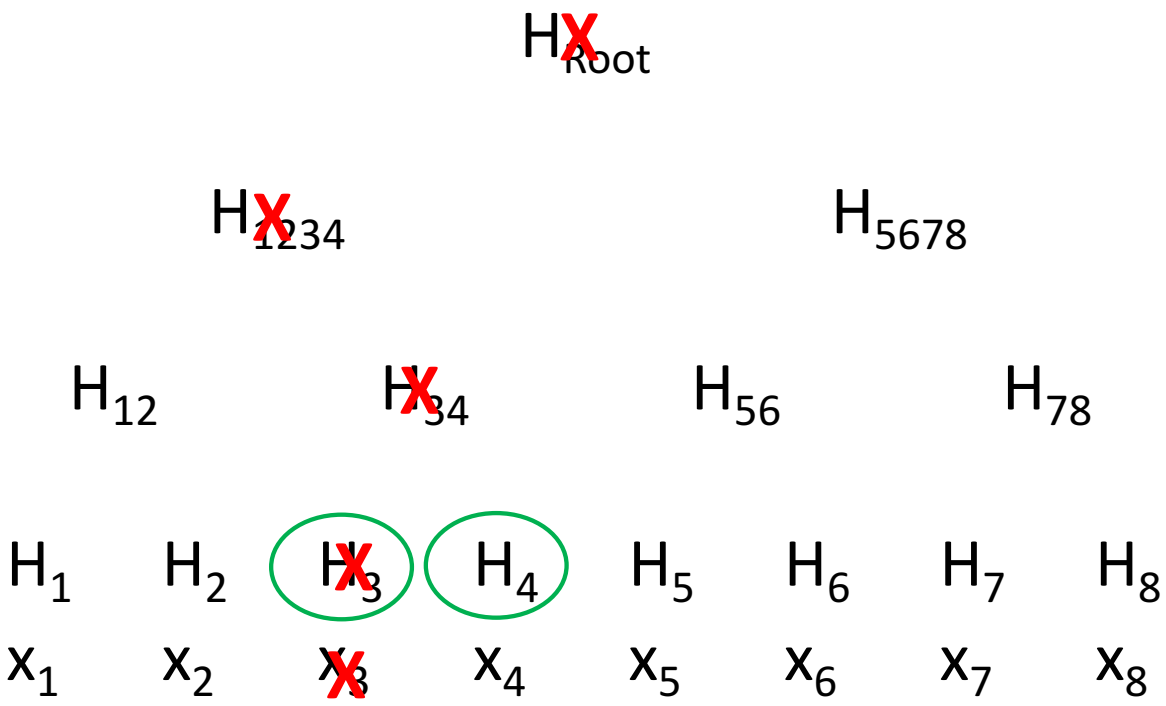
Hash Tree – Integrity Detection



Hash Tree – Integrity Detection



Hash Tree – Integrity Detection



Hash Tree

Verification efficiency

Scale exponentially with tree depth

Bitcoins

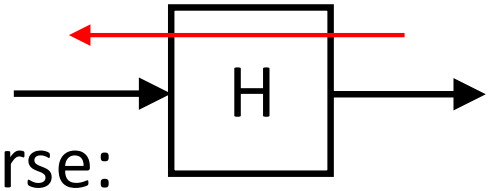
Verification with hash tree:

Merkle root (the root of hash tree) appended at the transactions

Bitcoins

Verification with hash tree:

Merkle root (the root of hash tree) appended at the transactions



Mining with hash reverse:

Bitcoin mining based on reversing hash function (SHA-256)