Key derivation functions

Key derivation: motivation

- Cipher algorithms require fixed dimension keys
  - 56, 128, 256… bits
- We may need to derive keys from multiple sources
  - Shared secrets
  - Passwords generated by humans
  - PIN codes and small length secrets
- Original source may have low entropy
  - Reduces the difficulty of a brute force attack
  - Although we must have some strong relation into a useful key
- Sometimes we need multiple keys from the same material
  - While not allowing to find the material (a password, another key) from the new key

Key derivation: purpose

- **Key reinforcement**: increase the security of a password
  - Usually defined by humans
  - To make dictionary attacks impractical

- **Key expansion**: increase/decrease a key length
  - Expansion to a size that suits an algorithm
  - Eventually derive other related keys for other purposes or algorithms
    - e.g. MAC, key hierarchies

Key derivation: approach

- Key derivation requires the existence of:
  - An initial key
  - A final key length
  - An item which makes the derivation unique
    - Salt, context or label
  - A difficult problem
    - Usually, a digest-based function
  - An optional chosen level of derivation cost
- Derivation cost alternatives
  - Computational difficulty
    - Transformation requires relevant computational resources
  - Memory difficulty
    - Transformation requires relevant storage resources
    - Limits attacks using dedicated hardware accelerators

Elementary key derivation

- Arguments:
  - secret - initial key
    - Provided by humans (password) or otherwise generated by computers (key)
  - salt - a random value
  - H - an adequate digest function

$$key = H(secret, salt)$$

- Advantages:
  - Key can have a large length, and be truncated to the adequate length
  - Two passwords will result in different keys
  - Finding the key will not lead to the password
- Issues:
  - Simple, enabling brute force/dictionary attacks

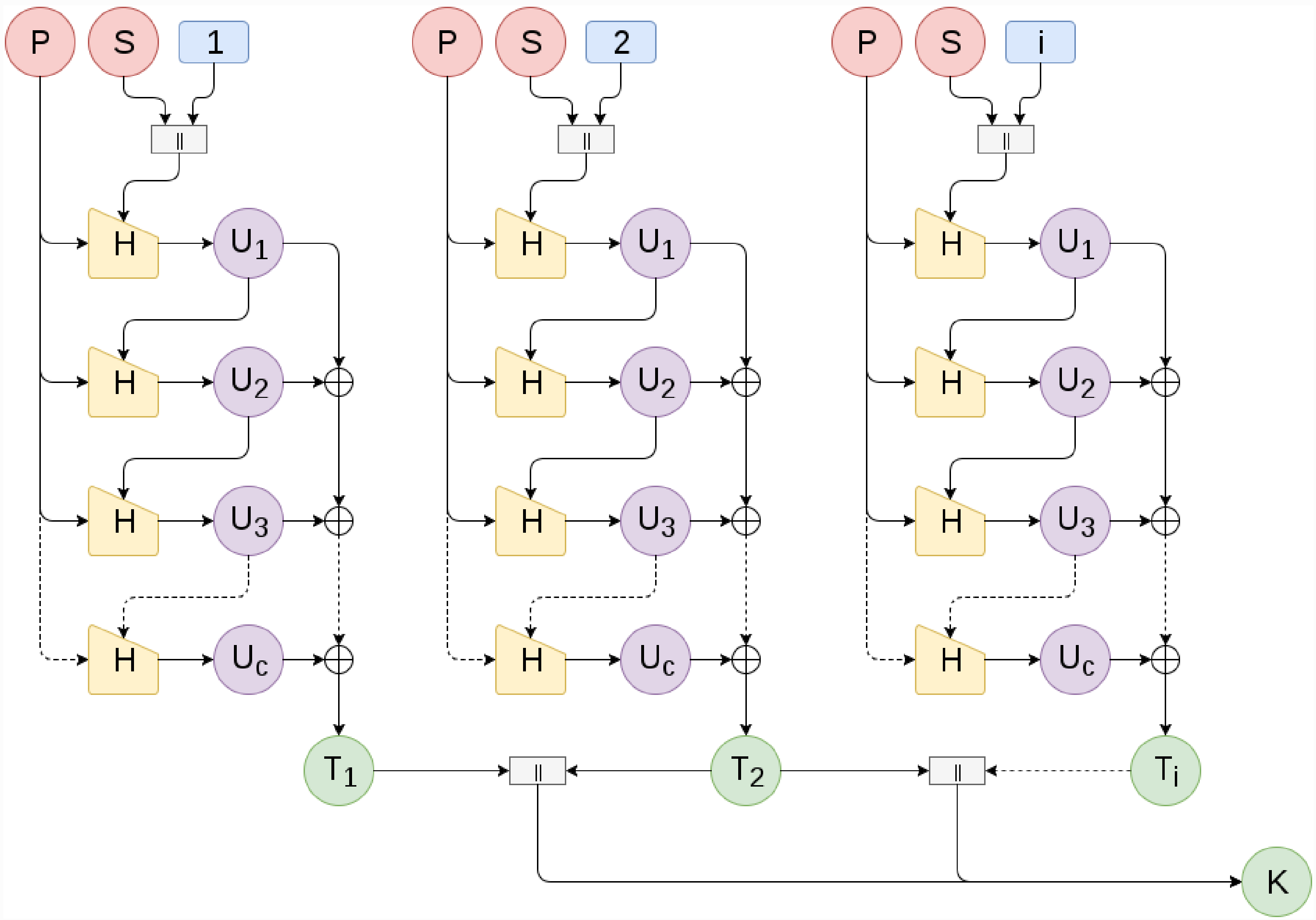
Complex key derivation

- Password Based Key Derivation Function (PBKDF2)
  - Produces a key from a secret with a chosen difficulty
  - secret – the initial secret
  - dim - the size of the resulting key
  - salt - a random value
  - PRF - Pseudo-Random-Function, typically a digest or MAC function
  - rounds - the computational cost (hundreds of thousands)

K = PBKDF2(PRF, secret, dim, salt, PRF, rounds)

- Operation:
  - Calculate rounds x dim' operations of PRF using the salt and password
  - dim' = $\lceil$ dim ÷ length of PRF output $\rceil$
  - Higher number of rounds increase the cost of secret discovery attempts

PBKDF2



P
S
1
||
H
U1
H
U2
H
U3
H
Uc
P
S
2
||
H
U1
H
U2
H
U3
H
Uc
P
S
i
||
H
U1
H
U2
H
U3
H
Uc
T1
||
T2
||
Ti
K

Other key derivation functions

- scrypt
  - PBKDF2-based
  - Adds an extra cost: size of intermediate results in memory
- Argon2
  - Blake2b-based
  - Also uses computational and memory costs