Solution to ensure TPM resistance to offline dictionary attack

Liqun Chen
HP Labs, Bristol

Mark D. Ryan
HP Labs, Bristol
University of Birmingham

Future of Trust 2008

June 2008

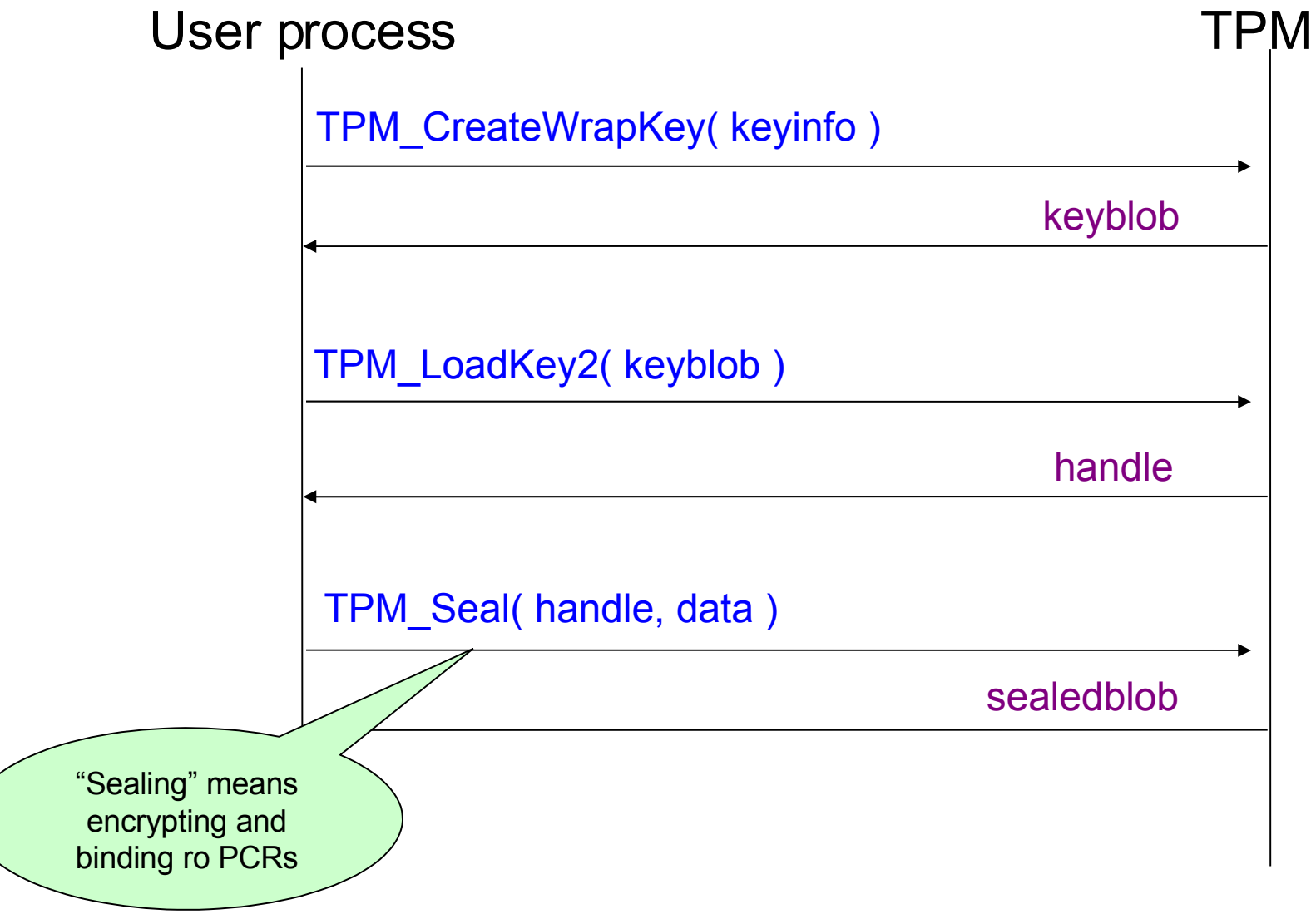
The Trusted Platform Module

- A hardware chip currently included in 100M laptops
  - HP, Dell, Sony, Lenovo, Toshiba . . . typically, high-end range
  - HP alone ships 1M TPM-enabled laptops each month
- Specified by the Trusted Computing Group
  - An industry consortium that includes Intel, HP, Microsoft, AMD, IBM, Sun, Lenovo. . .
  - and 130 other members
- Manufactured by many companies
  - Atmel, Broadcom, Infineon, Sinosun, STMicroelectronics, and Winbond

TPM functionality

- Secure storage
  - Creation of RSA keys (with private part known only to the TPM)
  - Encryption and decryption of user data with those keys
- Platform integrity reporting
  - “Measurement” and reporting of integrity of platform BIOS, disk MBR, boot sector, operating system and application software
- Platform authentication
  - Creation of *attestation identity keys (AIK)*, with anonymity guarantees (DAA)

TPM command message flow



TPM authData

- To each TPM object or resource is associated an authData value
  - A shared secret between the user process and the TPM
  - Think of it as a password that has to be cited to use the object or resource
  - authData is 20 bytes (160 bits)

- authData may be a weak (guessable) secret
  - May be based on a user-chosen password
  - E.g., in Microsoft Bitlocker,
    authdata = SHA-256(pin), truncated to 20 bytes

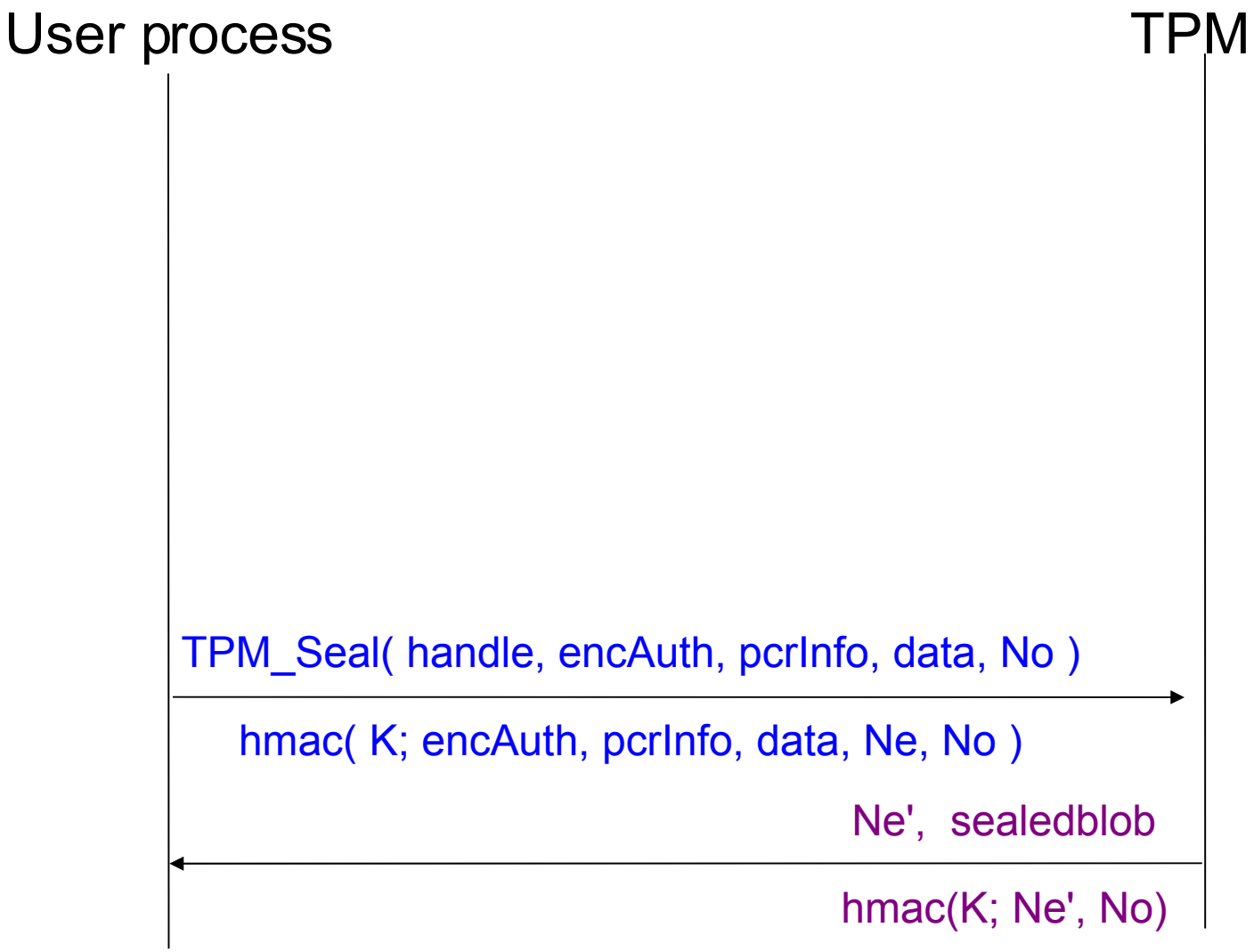
Weak authData

- The TPM resists online guessing attacks of weak authdata

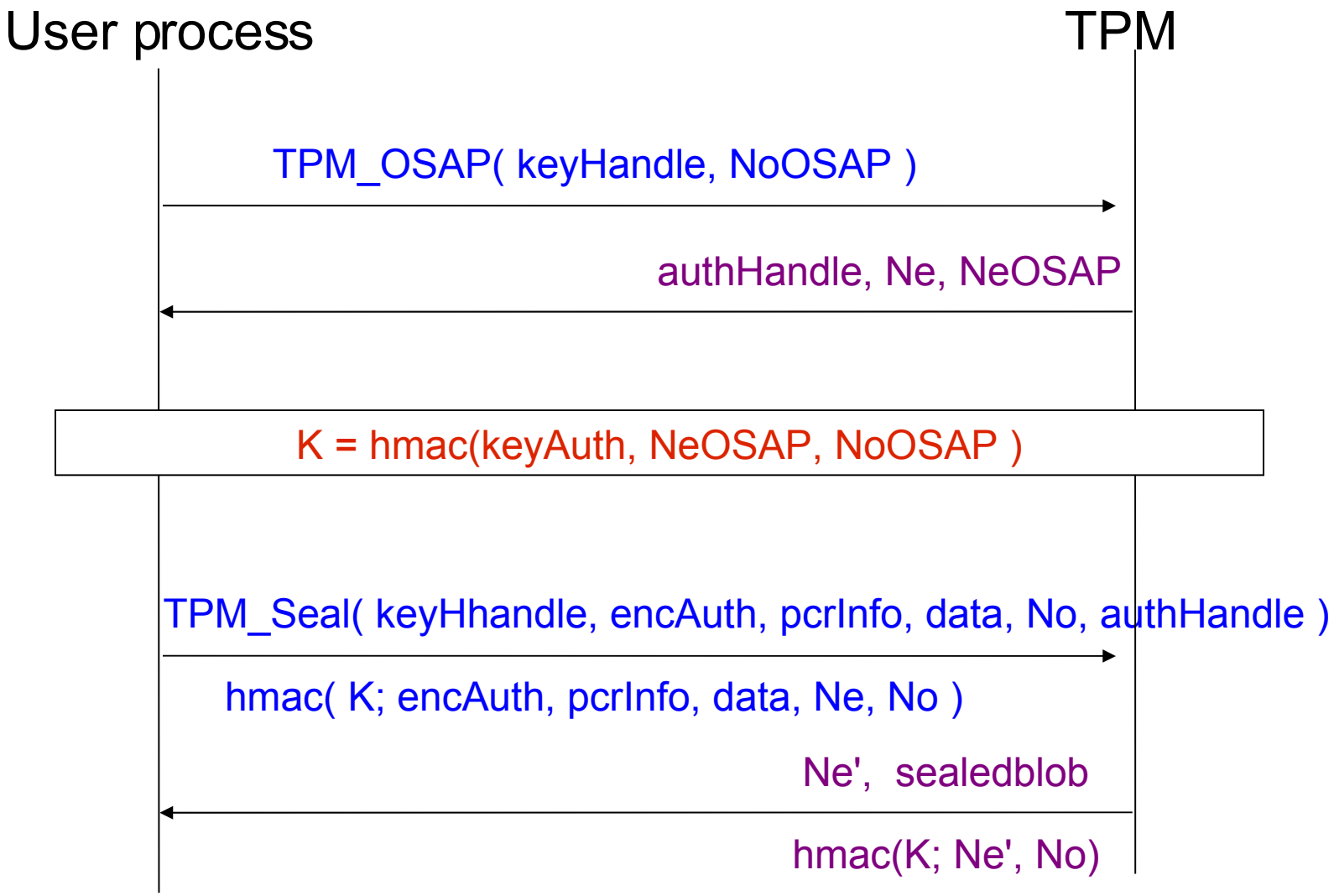
  "The decision to provide protections against dictionary attacks is due to the inability of the TPM to guarantee that an authorization value has high entropy. … Version 1.2 adds the requirement that the TPM vendor provide some assistance against dictionary attacks." [TPM Spec Part 1]

- The TPM should lock out a user that repeatedly tries wrong guesses
  - The details are left to the manufacturer

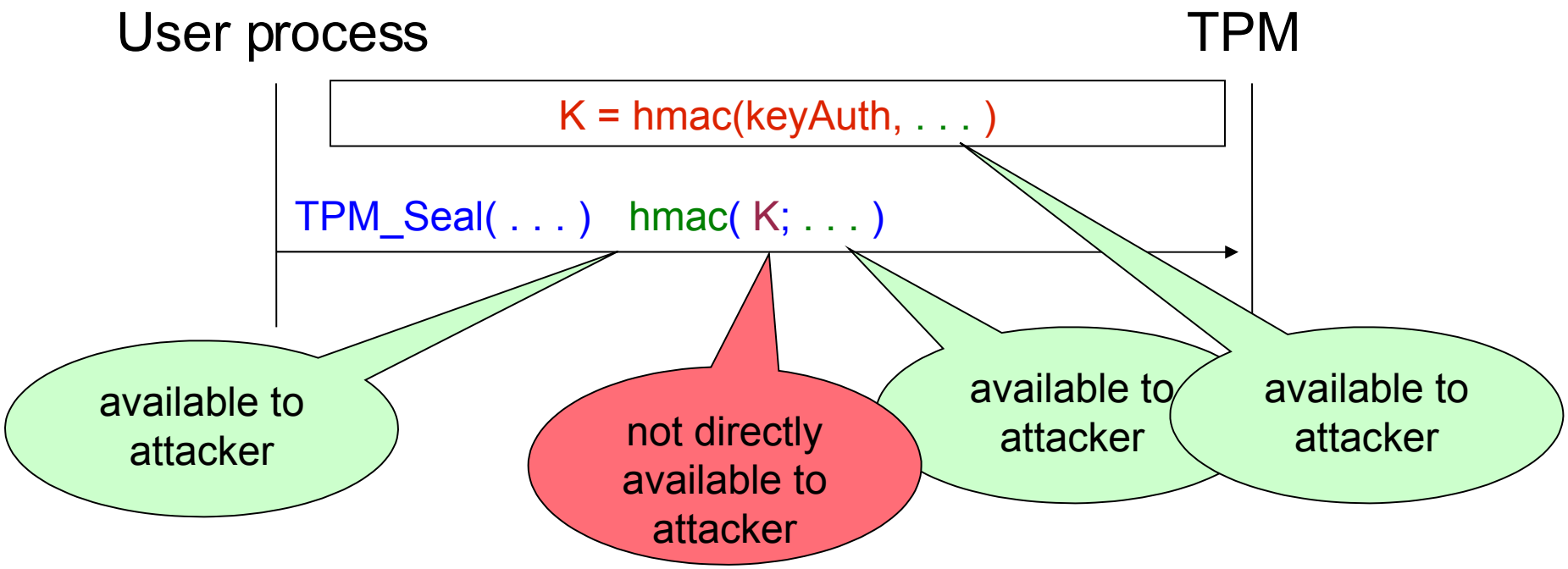
TPM_Seal


User process
TPM
TPM_Seal(handle, encAuth, pcrInfo, data, No)
hmac(K; encAuth, pcrInfo, data, Ne, No)
Ne', sealedblob
hmac(K; Ne', No)

TPM_Seal in more detail



The attack



The attacker can guess keyAuth, and can check his guess by reconstructing the hmac and comparing it with the hmac he observes.

Thus, the dictionary attack can be performed offline.

The resistance offered by the TPM doesn't help.

Objections 1

- The attack requires the attacker to see the message flow, and that is not possible.
  - Sometimes it is not possible (e.g., in pre-boot situations). But sometimes it is possible (e.g., remote usage).
  - If it was never possible, we would not have the HMACs.

- If the user choses strong authData, then the attack isn't possible.
  - That is correct. But the TPM spec mentions the possibility of weak authdata (e.g., hash of user password), and mandates resistance to online dictionary attacks.

Objections 2

- AuthData is supplemented by high-entropy nonce, e.g. in TPM_ChangeAuthAsymFinish
  - Nonce is sent in clear, so doesn't help
- TPM supports encrypted transport sessions
  - Encrypted transport sessions don't encrypt all the parts of the message on which this attack depends. The attack is still possible.
- Your attack requires the attacker to observe data flow. It won’t work if, for example, the attacker steals a switched-off laptop.
  - This is correct. Our attack requires the observation of traffic between the TPM and the user process.

Consequences

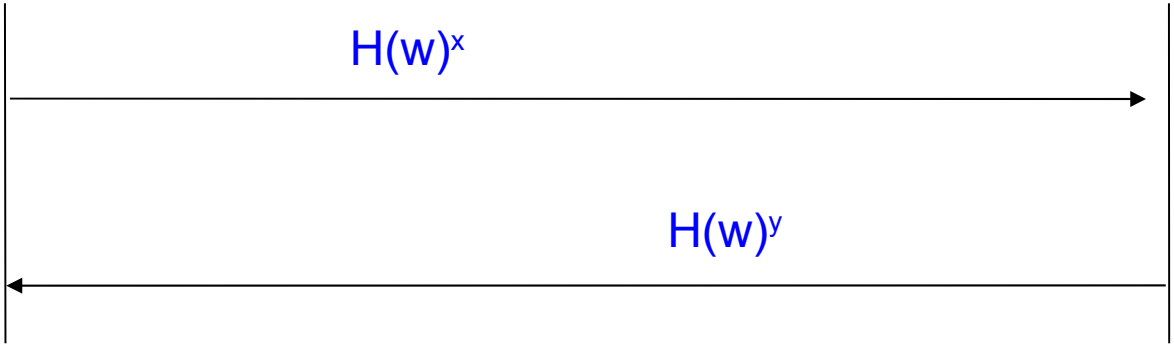
- As soon as authData is compromised, any new authData whose introduction is ADIP-protected by the compromised authData is also compromised
- Compromised authData means one user can impersonate another
  - including impersonating the TPM's owner
- Compromised authData means an attacker can impersonate the TPM
  - E.g., attacker can create its own key and forge response to TPM_CreateWrapKey. User binds with attacker's key. Attacker can unbind.

Solutions

- Aims
  - To prevent off-line dictionary attacks
  - To continue to allow authData to be weak secrets
  - To change the TPM spec as little as possible
    - minimise changes to the command structure
    - minimise requirement of extra memory in the TPM
    - minimise additional computational requirements

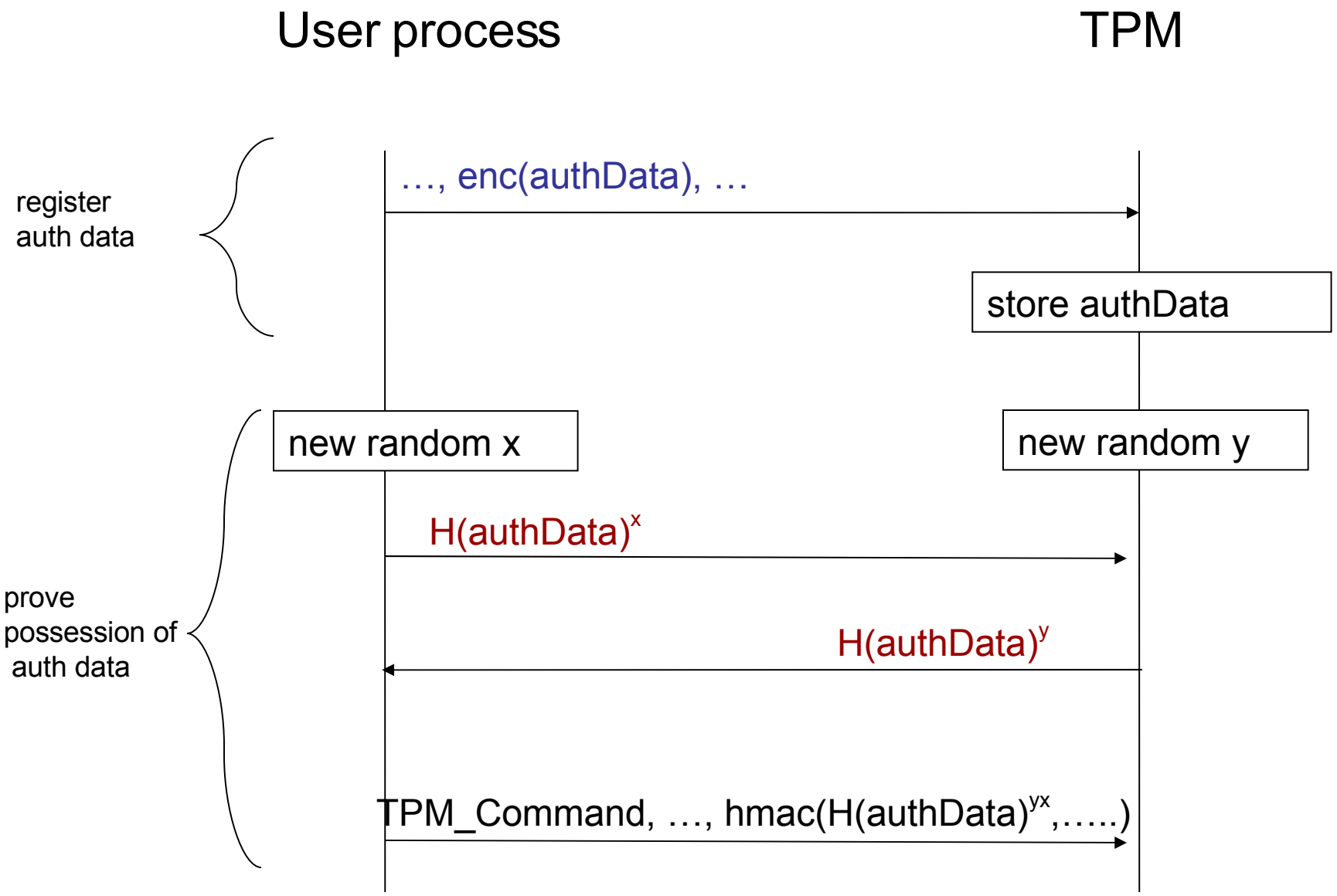
Simple Password Exponential Key Exchange (SPEKE) [Jablon 1996]

- A and B share a weak secret w
- Setup:
  - Let G be a finite field group of prime order q and prime modulus p, where q is at least 160 bits, and p is at least 1024 bits, such that $q \mid p - 1$.
  - Let H be a secure hash function $H : \{0, 1\}^* \rightarrow G$. We assume that the values q, p and the function H are known to A and B (they are not secrets)



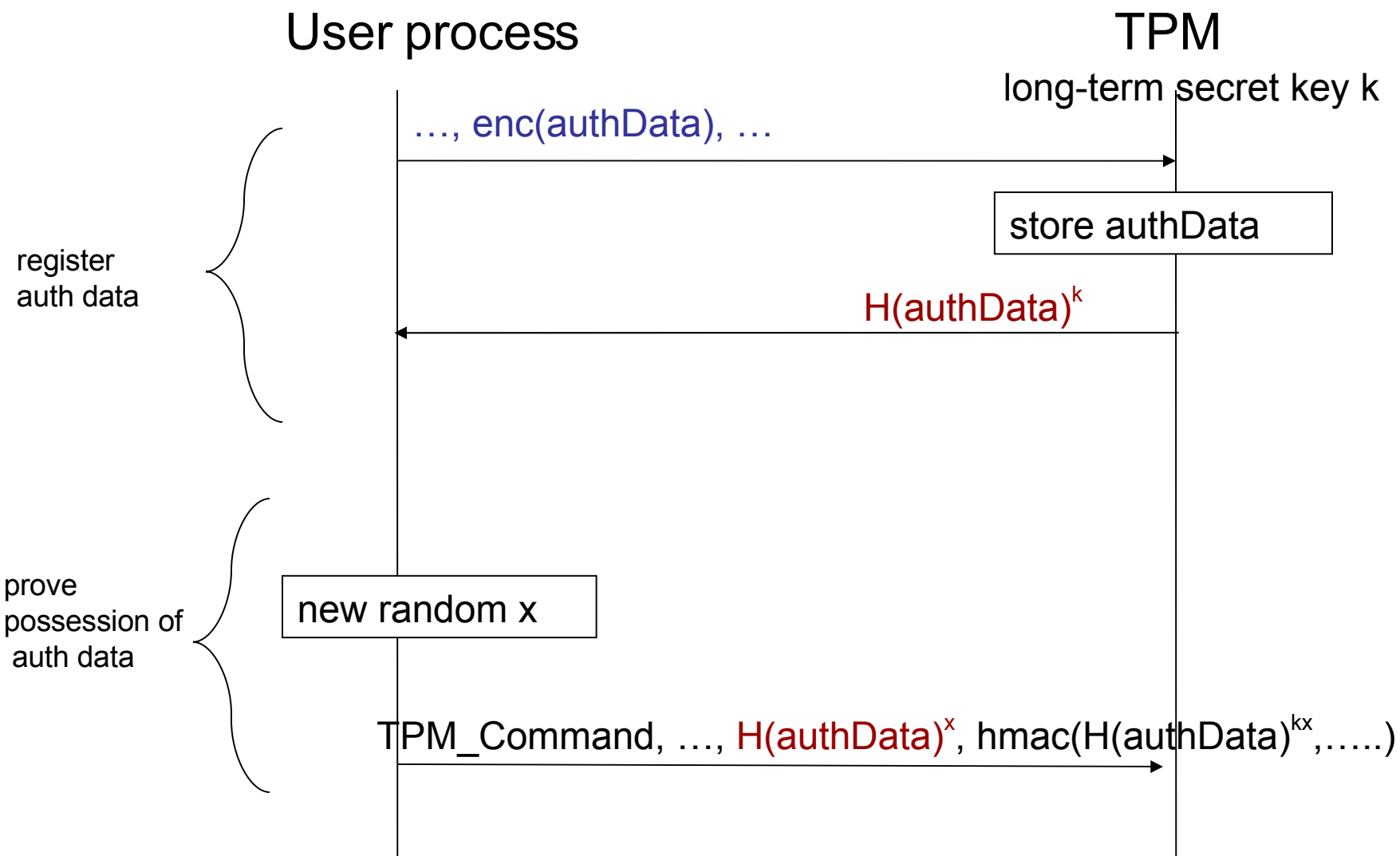
Now A,B share the strong secret $H(w)^{xy}$.

Solution 1: Use SPEKE



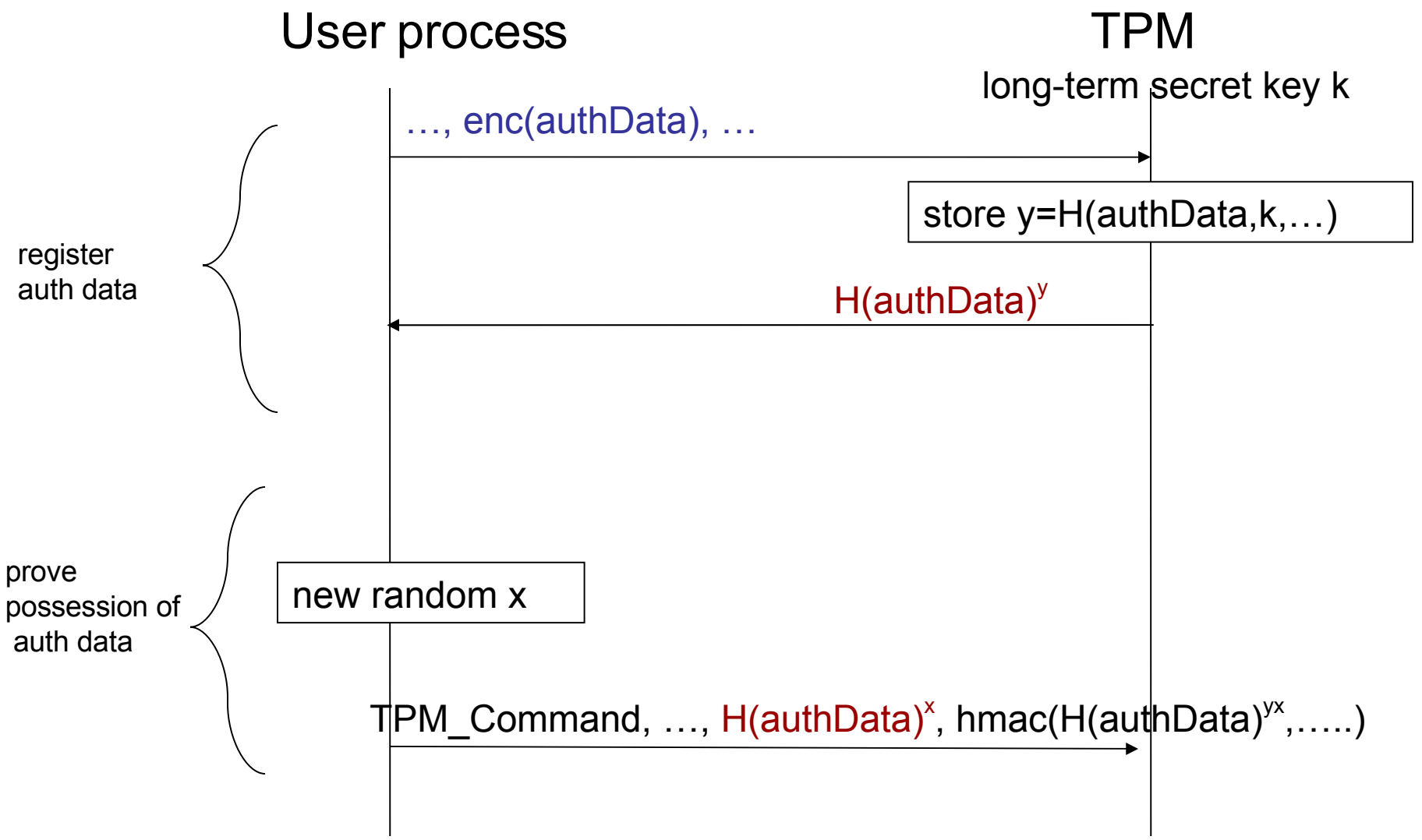
Solution 2: password-based key retrieval

reduce computation on TPM side



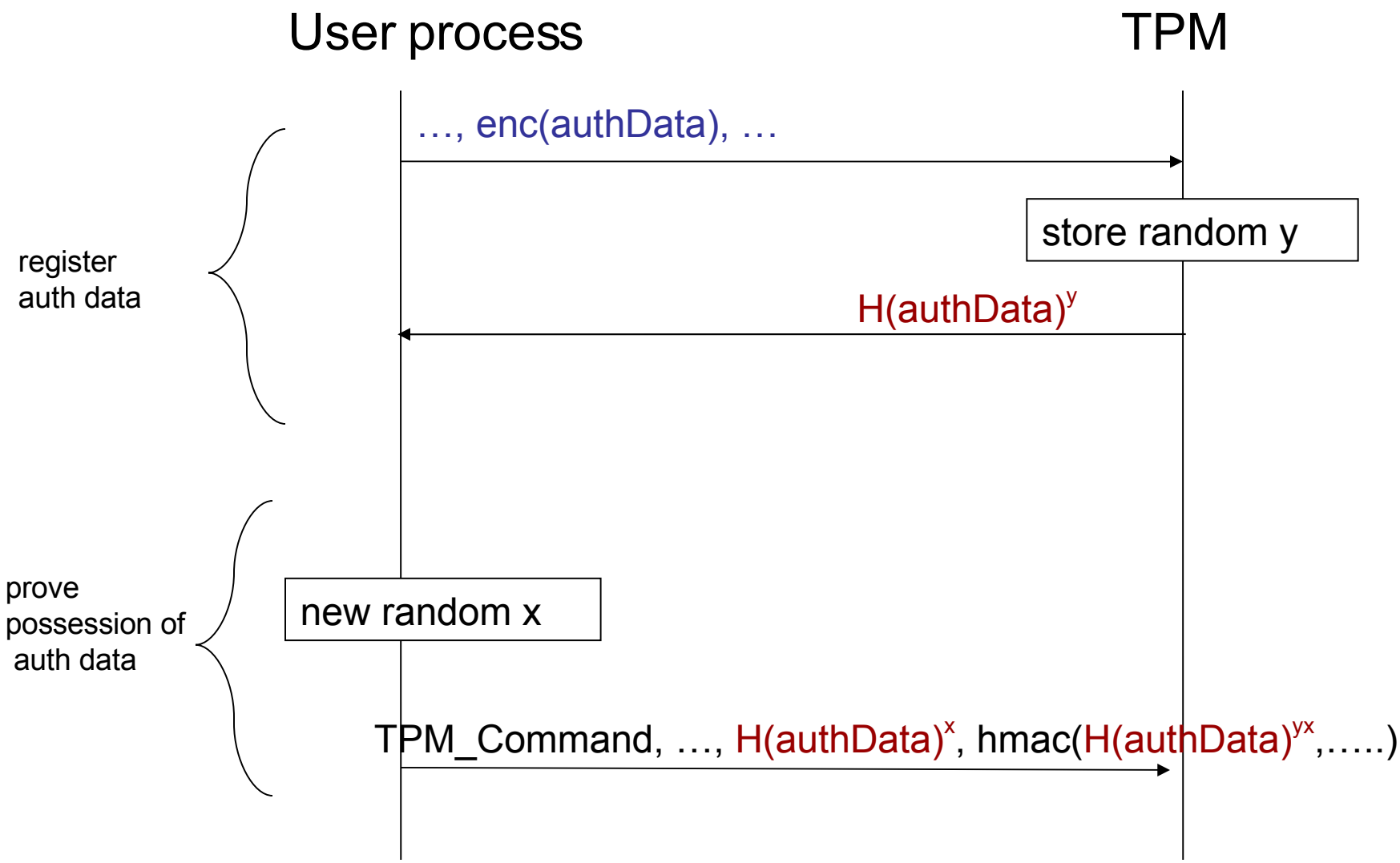
Solution 3: password-based proof of knowledge

avoid k being exposed



Solution 4: password-based proof of knowledge

avoiding long term key



Conclusions

- The solutions mean that weak data may be used in a way that does not expose it to offline guessing attacks.
- They are based on SPEKE, which in turn is based on Diffie-Hellman
- We are working with Trusted Computing Group to integrate solutions such as these into next version of TPM