SweetPAKE: Key exchange with decoy passwords

Afonso Arriaga, Marjan Skrobot, Peter Y.A. Ryan
SnT, University of Luxembourg

Luxembourg National Research Fund

ACM ASIACCS 2024
Singapore | 1 - 5 July, 2024

Stolen user credentials

- Modern password managers regularly monitor websites, forums, and marketplaces, where stolen user credentials are sold or published.
- Have I been pwned? https://haveibeenpwned.com/
    - 780 websites
    - 13B+ accounts
    - 42 new data breaches added in 2024, so far…
- Service providers must act promptly, but…
    - Businesses took an average of approximately **9 months** to identify and report a data breach [IBM Security, Cost of a Data Breach Report 2023]

Detecting leakage of user credentials

Decoy accounts

- Simple!
- Upgrade to the way users are authenticated is not required
- However, target attacks might remain undetected

username	password hash
mary	…e3b0c44298
fake	…fc1c149afb
john	…f4c8996fb9
mia90	…2427ae41e4

Detecting leakage of user credentials

Decoy passwords
[Juels & Rivest, CCS'13]

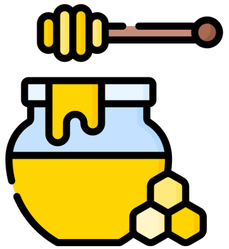
- Each user account is associated with one real password and multiple fake passwords (i.e. **honeywords**).
- The index of the real password os each user is stored in a separate device, i.e. the **honeychecker**. The goal of this device is to raise an alarm whenever a user used a honeyword as their password.
- Multiple methods to generate honeywords that are difficult to distinguish from real passwords have been proposed. The authors call this property **flatness**.

username	pwd1	pwd2	pwd3
mary	123456	qwerty	password
philip	1q2w3e	password1	1qaz2wsx
john	1qaz2wsx	qwertyuiop	1q2w3e4r5t
mia90	ronaldo1	princess	status

Detecting leakage of user credentials

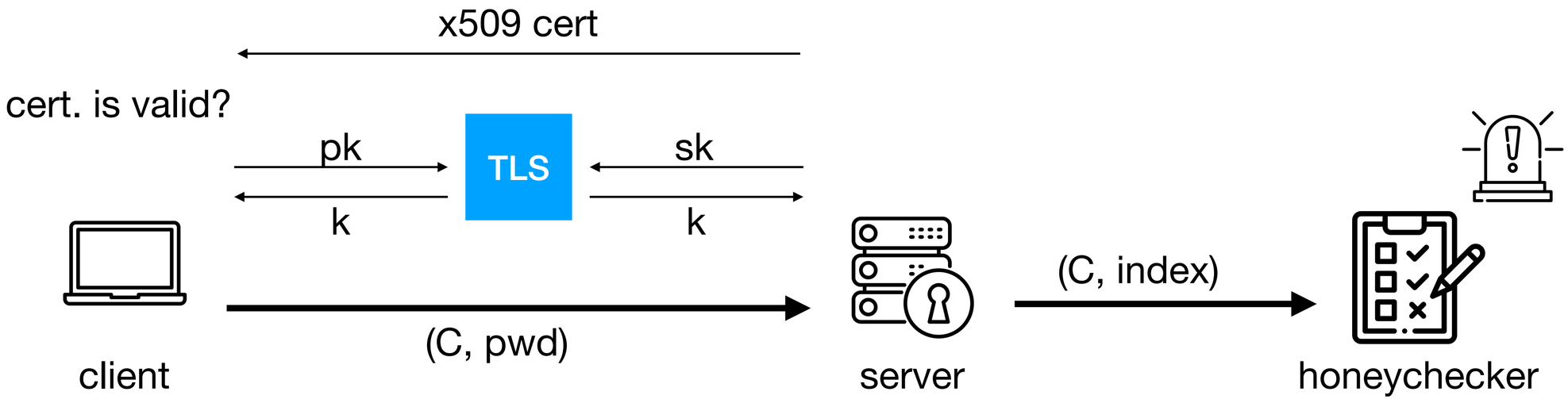
Decoy passwords
[Juels & Rivest, CCS'13]

How to generate honeywords?



- **Tweaking** (e.g. last 3 symbols): "BG+7y45" → "BG+7**q03**", "BG+7**m55**", "BG+7**y45**", …
- **Password-model**: "nice72fit" → W4 | D2 | W3 → "gold51nut", "asia24ccs", …
- **Take-a-tail**: "merlion" → "merlion**972**" → "merlion**122**", "merlion**404**", "merlion**590**", …
- **Random pick**: "ueF9RiDd", "**ic80U3Hp**", "N0HemkrG", "kdMEz9re" → "**ic80U3Hp**"

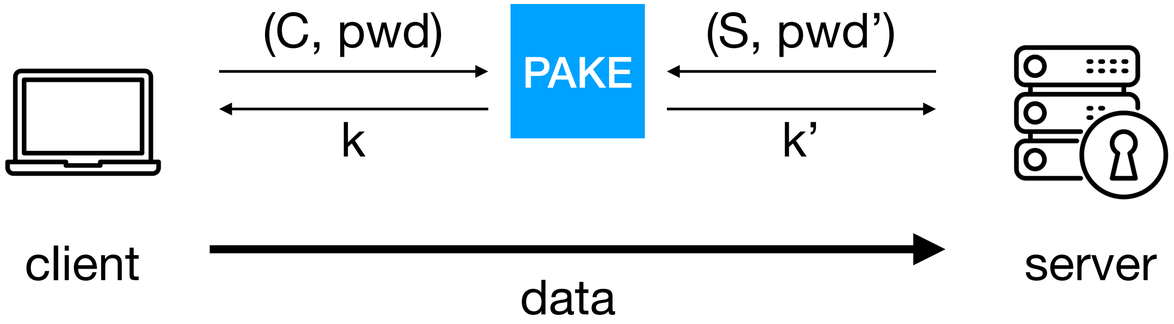
Honeywords setup



- User credentials are sent in “clear” over the server-authenticated secure channel.
- Requires trust in a PKI to validate server certificate.

Password Authenticated Key Exchange

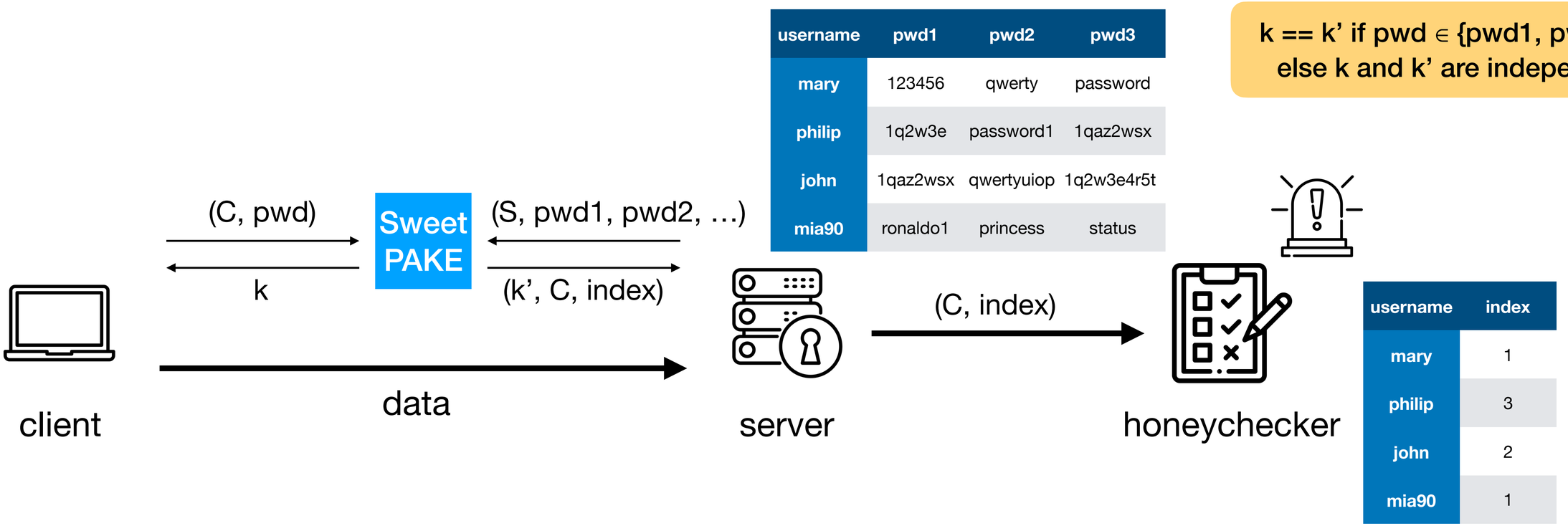
- Cryptographers know how to mutually authenticated server and client with only the password for over 30 years [Bellovin&Merritt, S&P'92]
- PAKE has seen adoption in the past decade, e.g. ePassports, WiFi WPA3, password managers, Apple iCloud Keychain, Proton Mail, etc.
- Benefits of PAKE:
  - No PKI attacks
  - No phishing attacks



k == k' if pwd == pwd'
else k and k' are independent

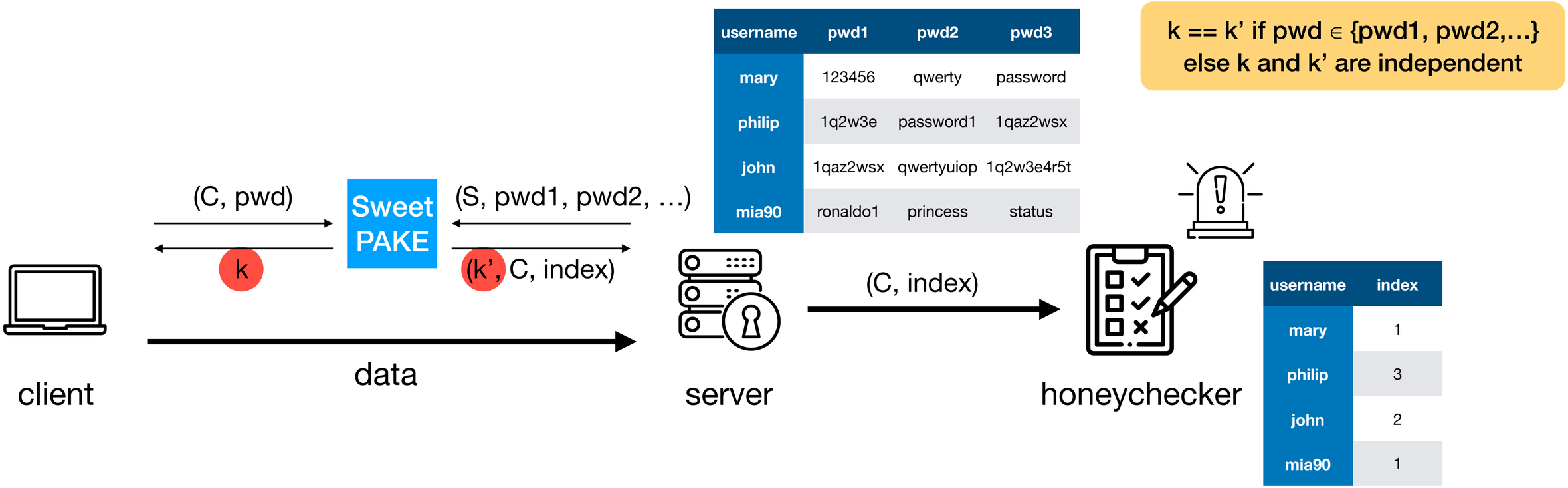
PAKE with decoy passwords

1. *What security guarantees such protocol provides?*
2. *How to build a PAKE-style protocol that admits decoy passwords?*



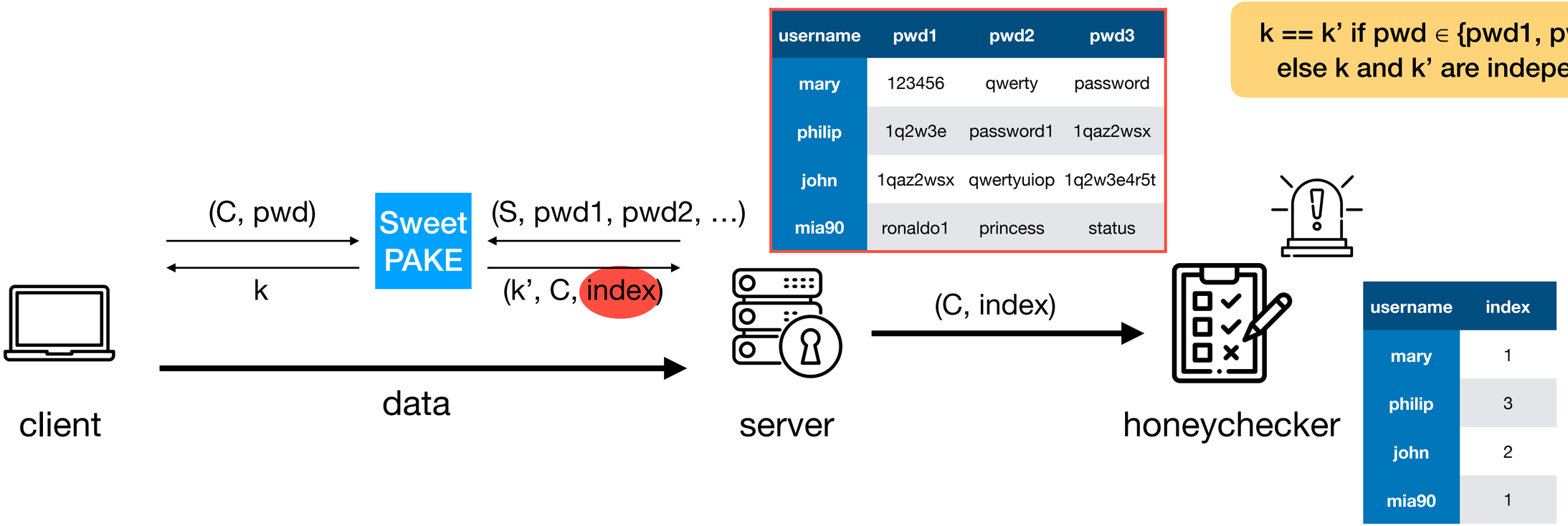
Security properties of SweetPAKE

1. **<u>Indistinguishabilty of session keys</u>**, i.e. similar to PAKE.



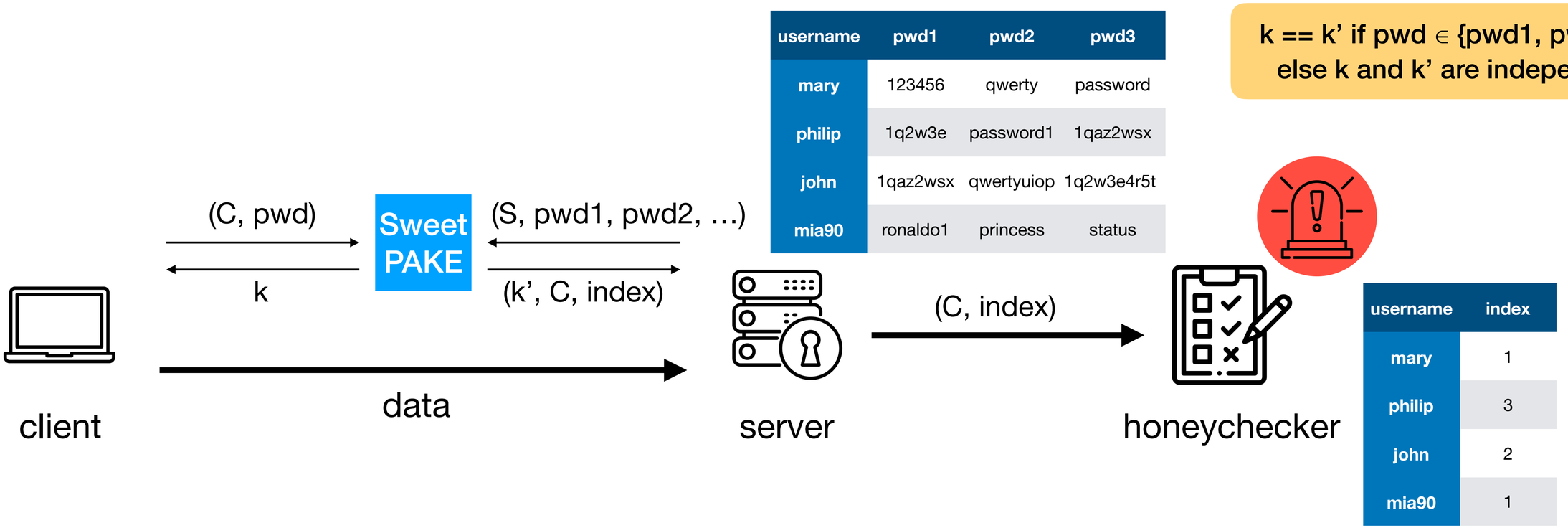
Security properties of SweetPAKE

1. Indistinguishabilty of session keys, i.e. similar to PAKE.
2. **Sugardword indistinguishability**, i.e. difficulty of guessing the sugarword among sweetwords, after a password file leak, even for adversaries that have some control over the network.



Security properties of SweetPAKE

1. Indistinguishabilty of session keys, i.e. similar to PAKE.
2. Sugardword indistinguishability, i.e. difficulty of guessing the sugarword among sweetwords, after a password file leak, even for adversaries that have some control over the network.
3. **False alarm protection**, i.e. the difficulty of an adversary triggering the alarm without the password file being leaked.

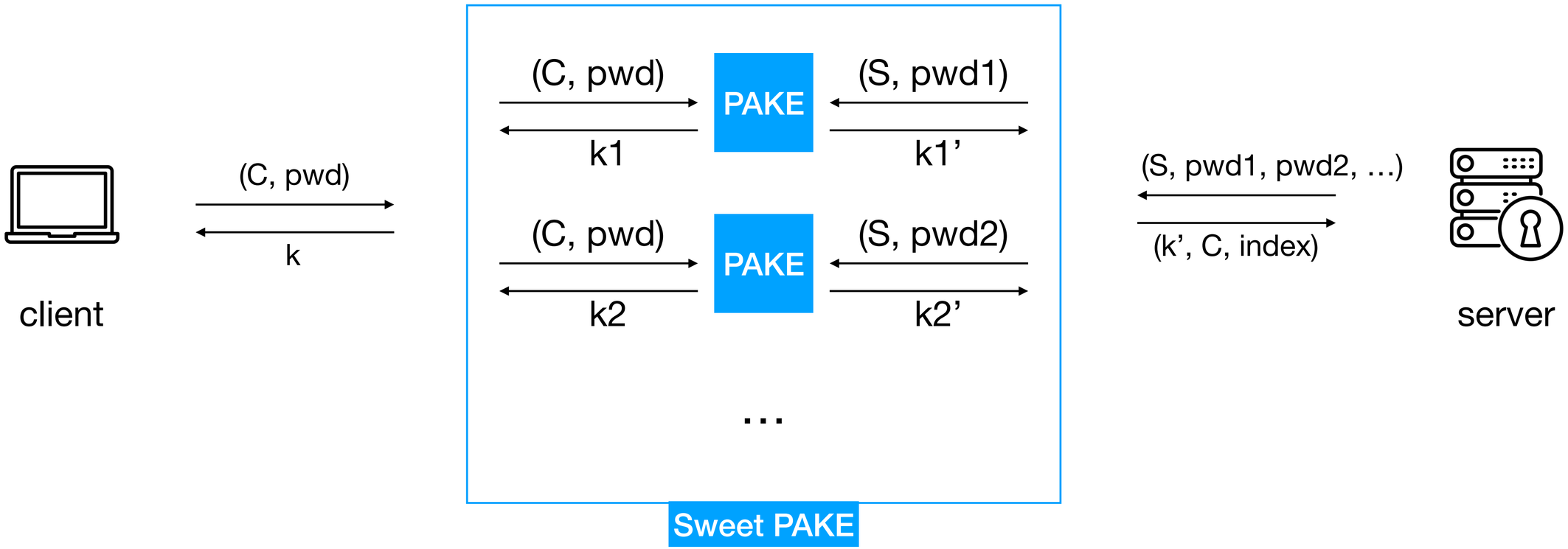


How to model security?

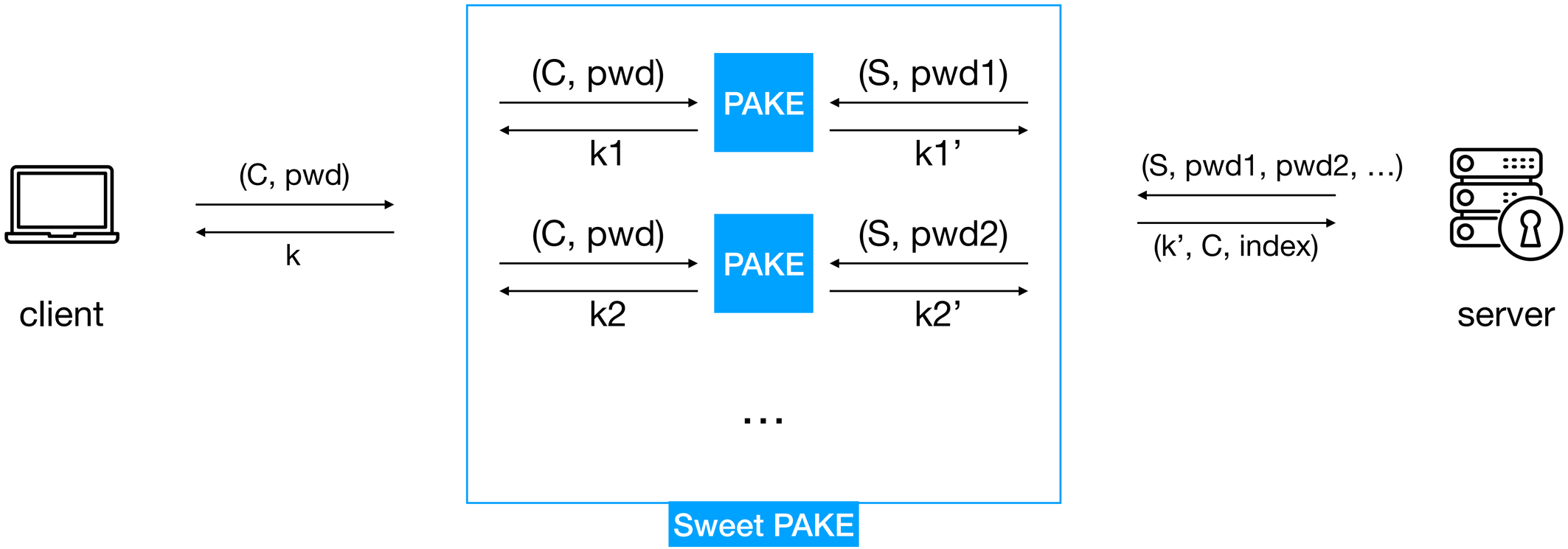
Two approaches:

1. Game-based security [BPR00, AFP05]:
    - Simple and generally easier to understand.
    - Challenger samples passwords uniformly at random.
    - We extend the models with a *Leak*(C,S) oracle to model a compromise of the password file.
2. Universal Composability (UC) [CHK+05]:
    - Gold standard for PAKE security.
    - Passwords are arbitrarily chosen by the environment Z. Captures password correlations, which is crucial depending on the honeyword generation method.
    - The ideal functionality reflects the security aspects of SweetPAKE.
    - If the ideal functionality does not leak the index of the sugarword, it captures sugarword indistinguishability.

The naive approach



The naive approach



1. Can we optimize the client computation and reuse the same key share in each of the *n* instances?
2. No sugarword indistinguishability!

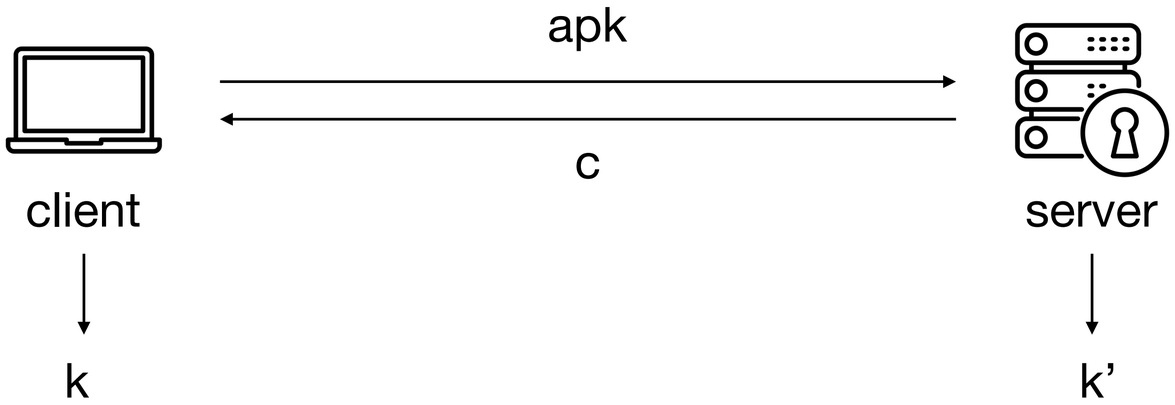
Password Authenticated PKE

Reusing the first flow of messages generally breaks PAKE security …but we can use PAPKE instead!

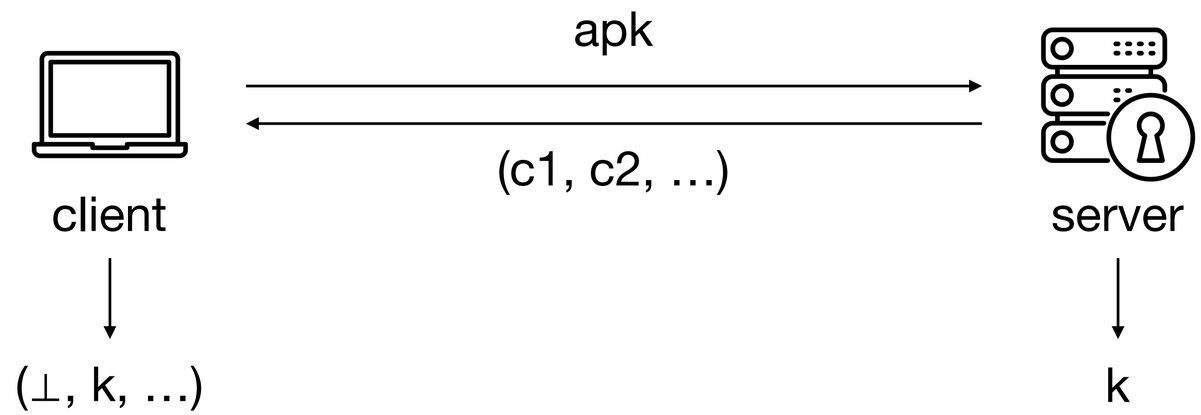
PAPKE [BCJ+19]:

1. KGen(pwd) → Outputs secret key **sk** and password-authenticated public key **apk**.
2. Enc(apk, pwd', m) → Outputs a ciphertext **c**.
3. Dec(sk, c) → Outputs **m** if pwd == pwd', ⊥ otherwise.

PAPKE-2-PAKE compiler (simplified):

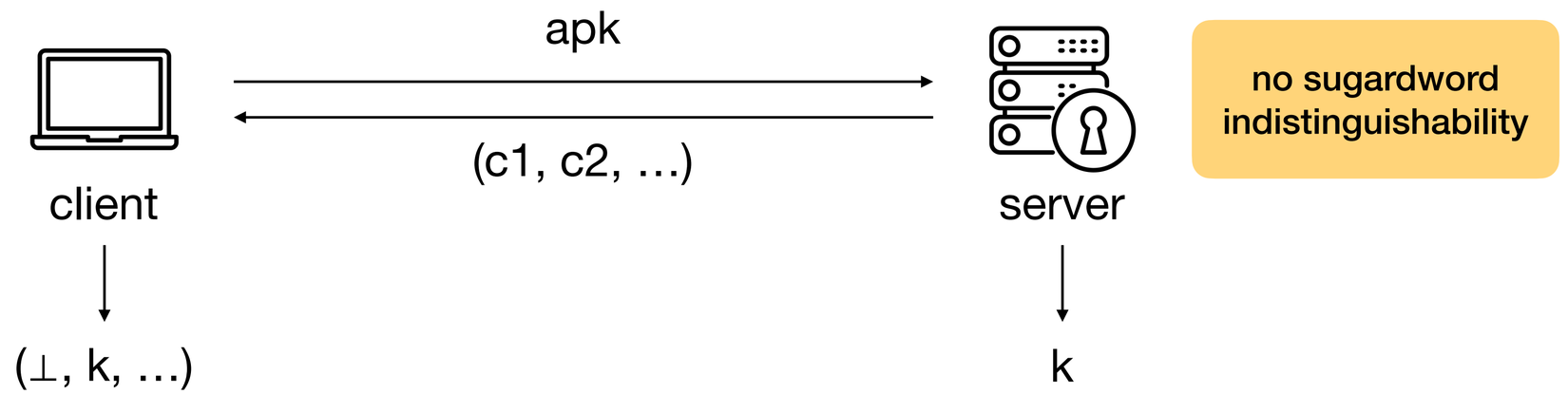


BeePAKE protocol



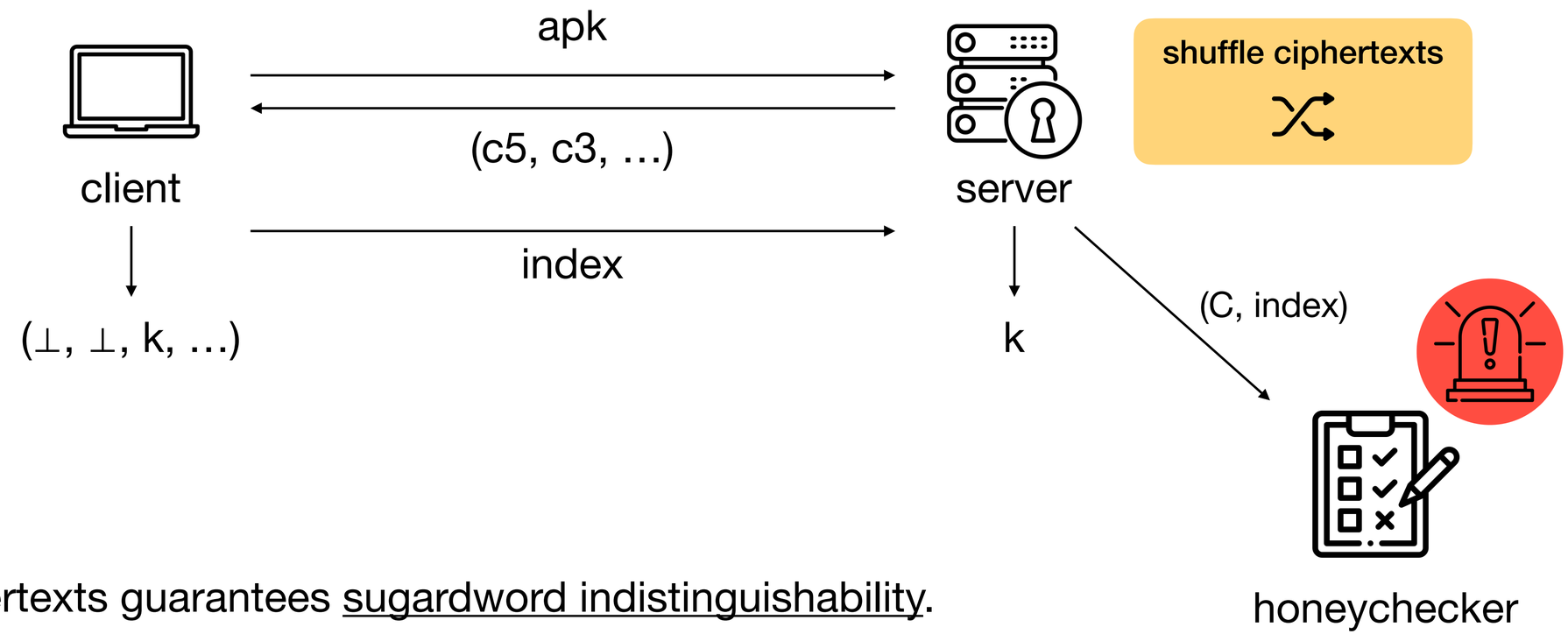
- Client only need to send apk once.
- Server encrypts key k *n* times, under each sweetword (candidate password).

BeePAKE protocol



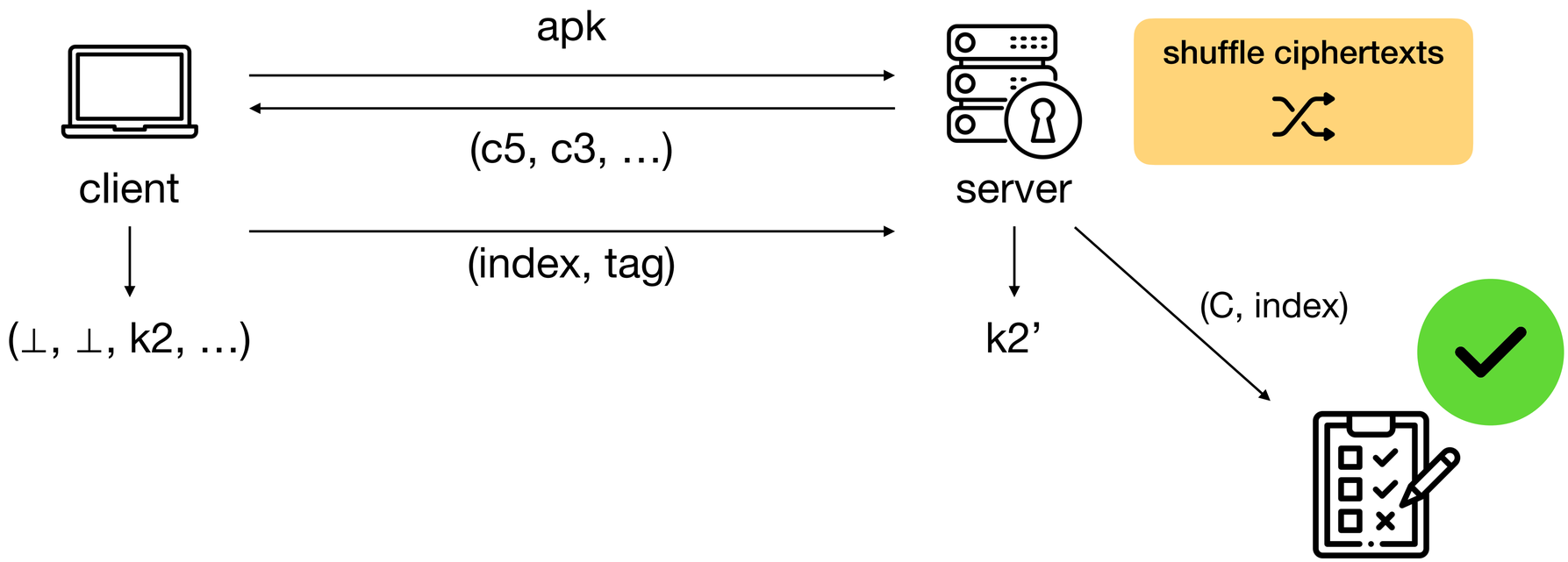
- Client only need to send apk once.
- Server encrypts key k *n* times, under each sweetword (candidate password).
- An attacker can easily figure out the index of the sugarword by interacting with the client before the password file leak, e.g. sending (c1, c1, …) to the client reveals if the sugarword is in index 1.

BeePAKE protocol



- Shuffling the ciphertexts guarantees sugardword indistinguishability.
- We have to be careful regarding key confirmation, to avoid false alarms.

BeePAKE protocol



- Shuffling the ciphertexts guarantees sugardword indistinguishability.
- We have to be careful regarding key confirmation, to avoid false alarms.
    - Encrypt a different key under each ciphertext and derive a tag to offer false alarm protection.

Conclusions

- A PAKE-style protocol that admits decoy passwords is useful to put in place as a password file leakage detection mechanism.
- We identified and modelled the desired security properties using both a game-based approach and within the UC framework.
- We propose BeePAKE: a simple protocol that builds on top of PAPKE and offers (1) session key indistinguishability; (2) sugarword indistinguishability; (3) false alarm protection.
- We observe that the SweetPAKE notion is closely related to the Oblivious PAKE notion [Kiefer&Manulis, IS'15], with the roles of client and server reverse.
    - Our UC definition unifies both primitives and captures arbitrary passwords correlations, which is relevant in both settings.
    - Our protocol offers a simple and modular approach.
- We instantiated BeePAKE with PAPKE-IC-DHIES and benchmarked it against the naive approach of running *n* SPAKE2 protocols in parallel. As expected, the computational cost imposed on the client side is considerably lower because the client only needs to send apk once (the cost is about ~50% lower for n=20).

Thank you for your attention!

afonso.delerue@uni.lu