MEGA: Malleable Encryption Goes Awry

Miro Haller[1,2]
mhaller@ucsd.edu

Matilda Backendal[1]
mbackendal@inf.ethz.ch

Kenny Paterson[1]
kenny.paterson@inf.ethz.ch

[1]ETH Zurich
[2]University of California San Diego

44th IEEE Symposium on Security and Privacy, May 2023

Who is MEGA?

"MEGA does not have access to your password or your data."
https://mega.io/security (2022)

The largest end-to-end encrypted cloud storage:

- 280M+ accounts
- 140B+ files
- 10M+ active users
- 200+ countries

src: https://mega.io/about (05/2023)

Attack Teaser

5 attacks

allow a malicious cloud provider to

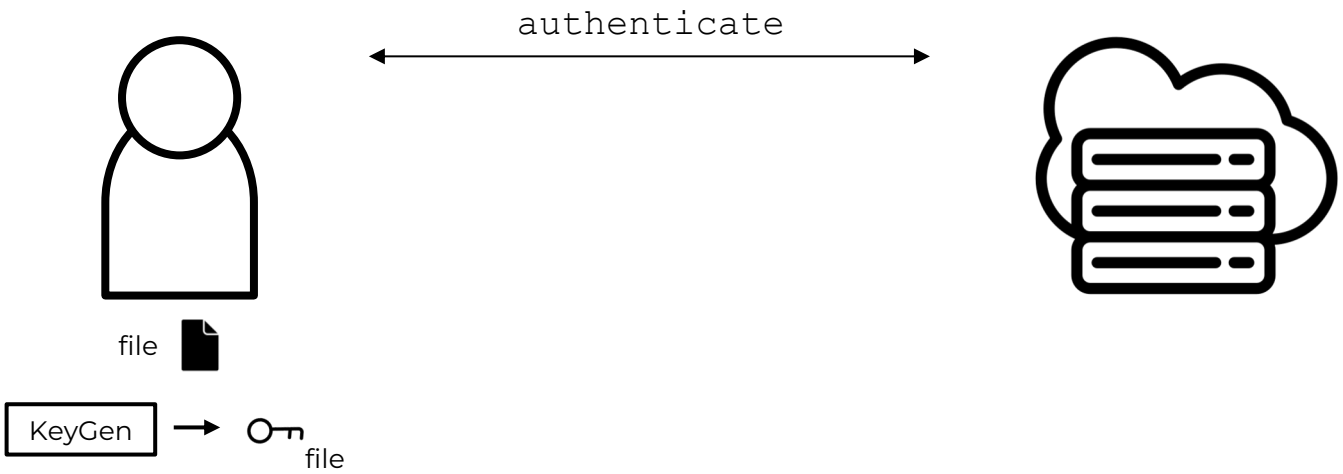
- ✓ Break authentication
- ✓ Read user files
- ✓ Upload new files

Cryptographic design of MEGA*

*strongly simplified

File upload*

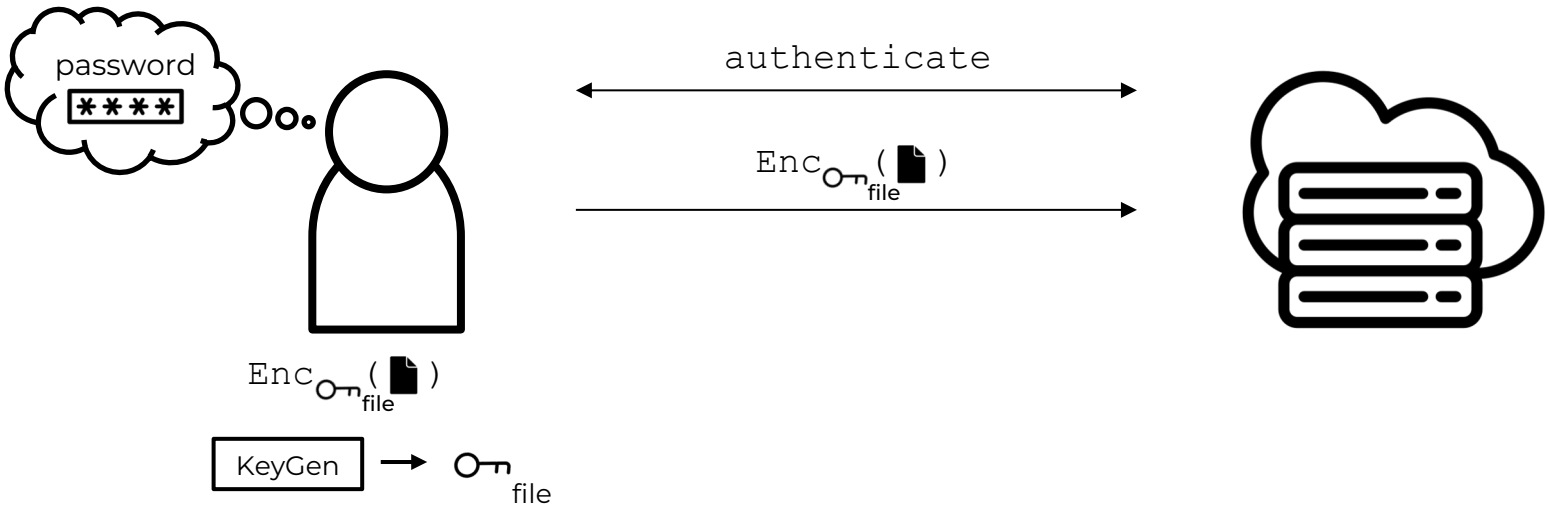
Upload locally encrypted file and key.



*strongly simplified

File upload*

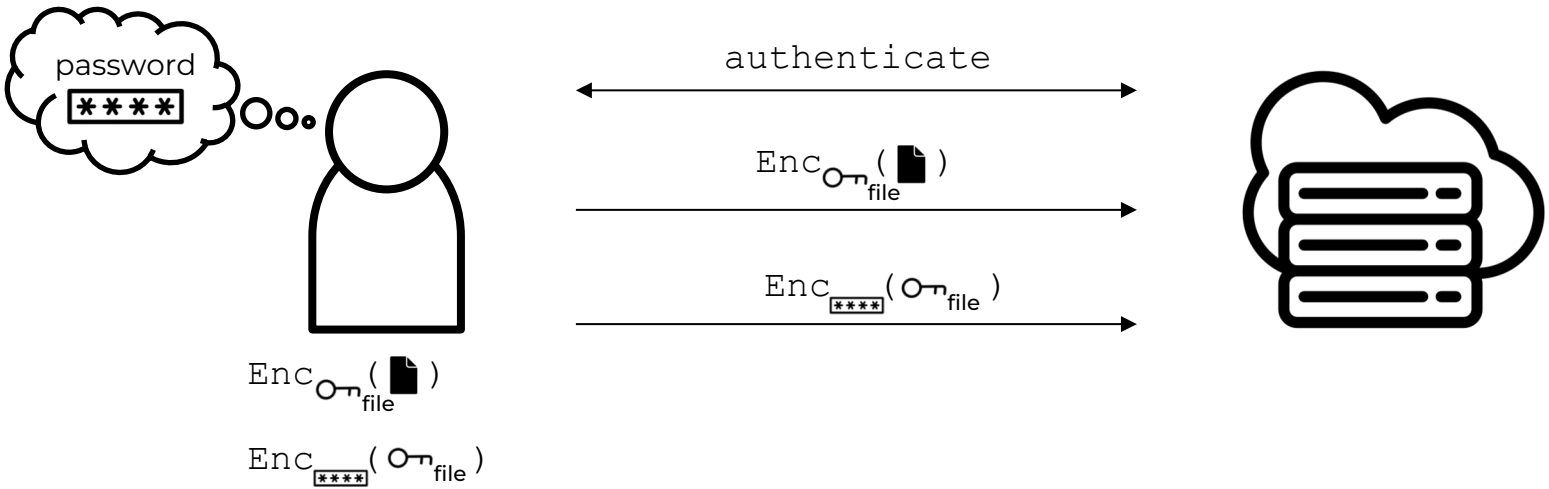
Upload locally encrypted file and key.



*strongly simplified

File upload*

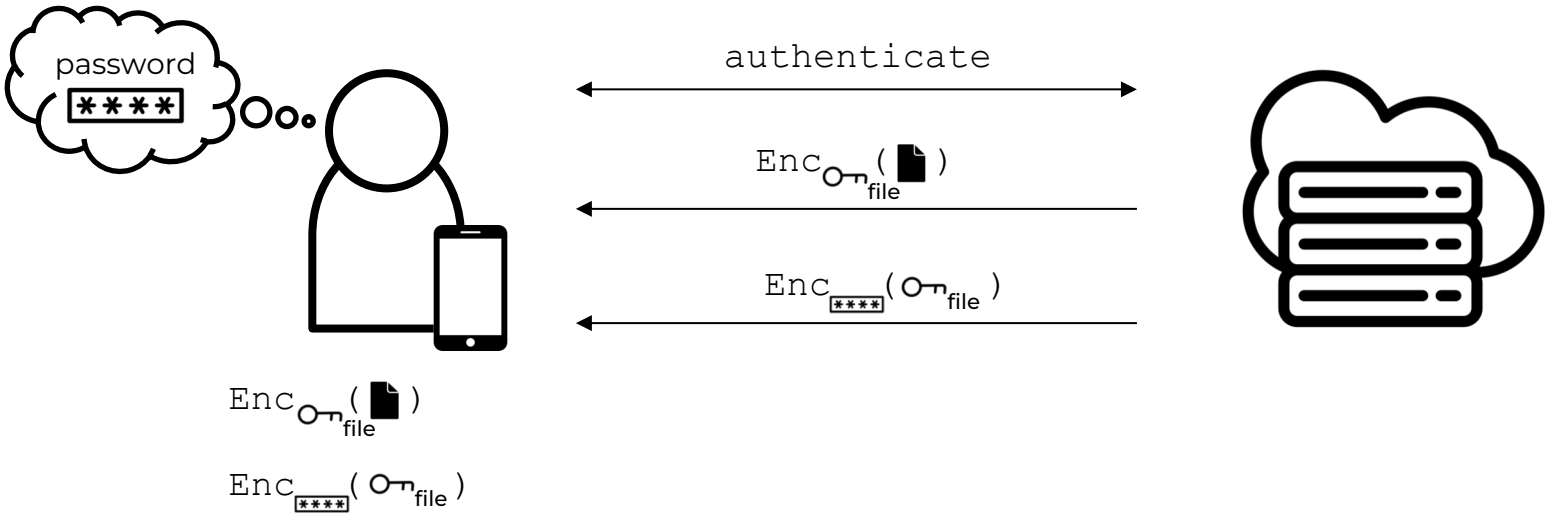
Upload locally encrypted file and key.



*strongly simplified

File download*

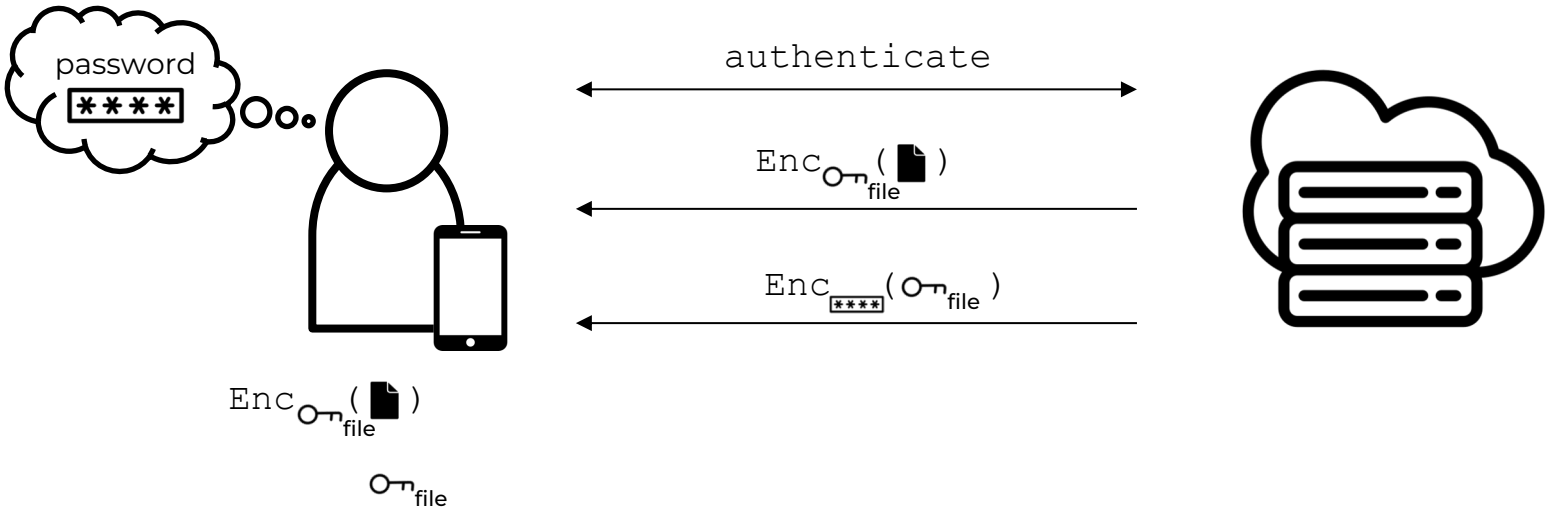
Download encrypted file and key, decrypt locally.



*strongly simplified

File download*

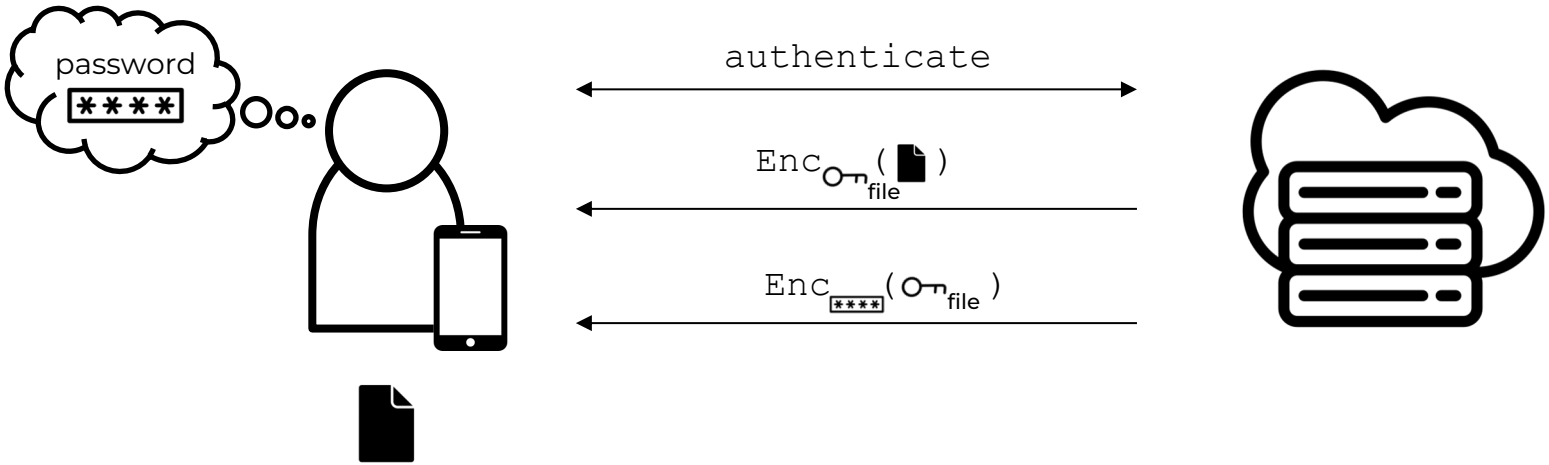
Download encrypted file and key, decrypt locally.



*strongly simplified

File download*

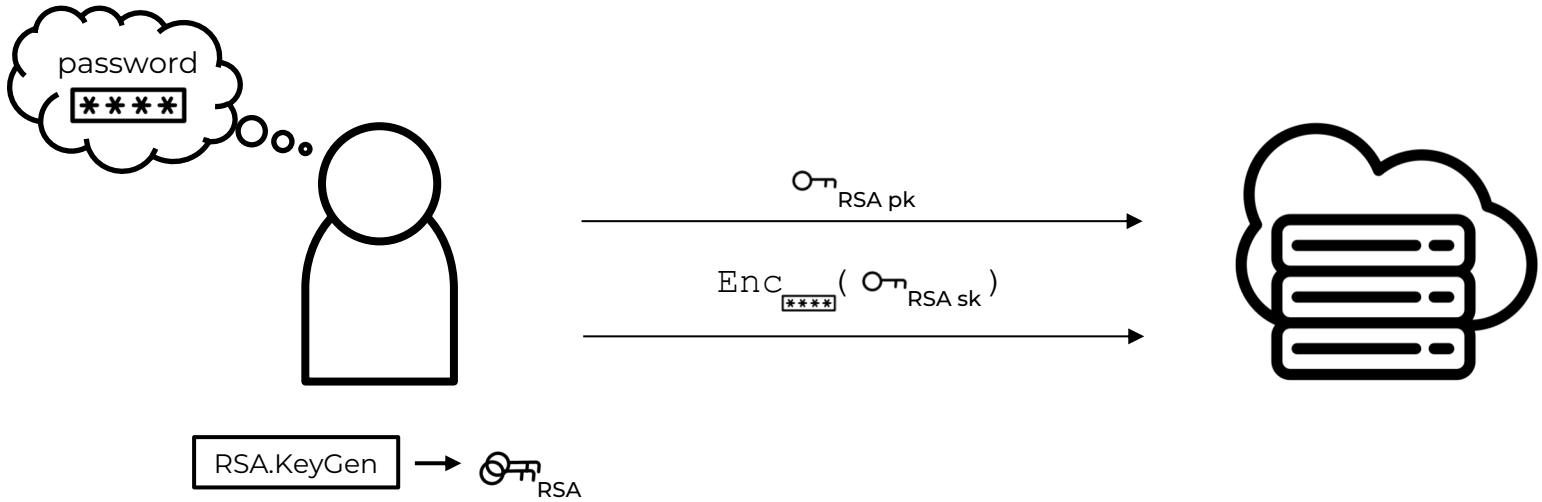
Download encrypted file and key, decrypt locally.



*strongly simplified

Registration*

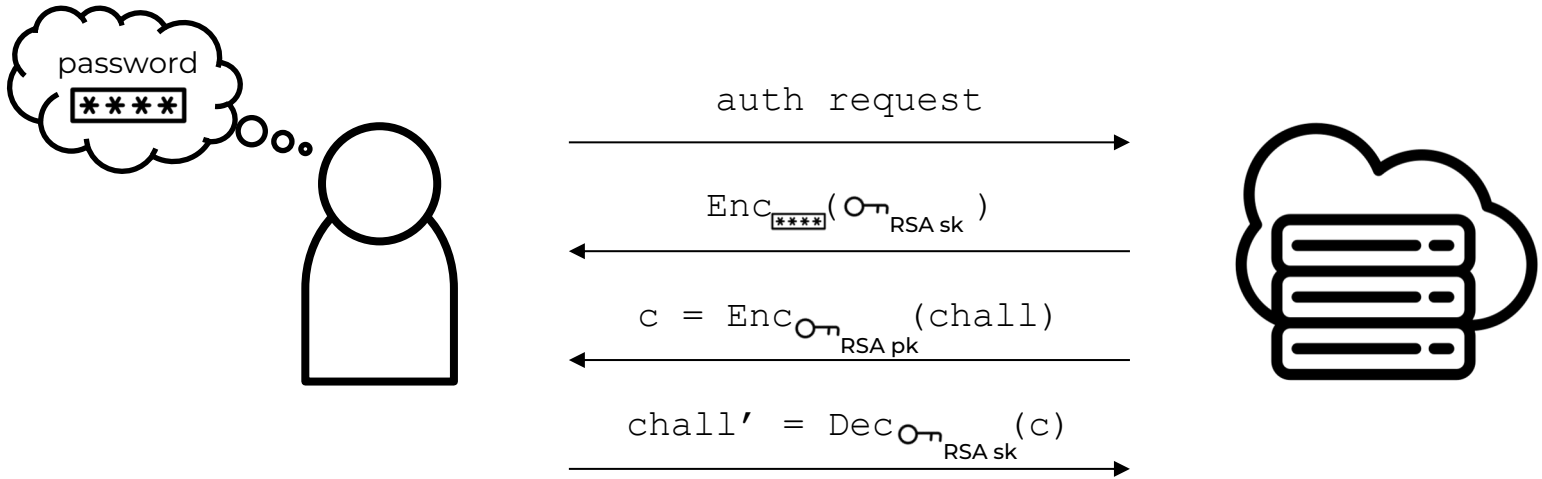
Generate and upload RSA secret key for authentication.



*strongly simplified

Authentication*

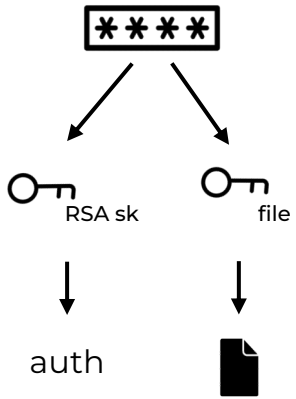
Client proves knowledge of password in challenge-response protocol.



*strongly simplified

Key hierarchy*

Two types of keys protected by the password.

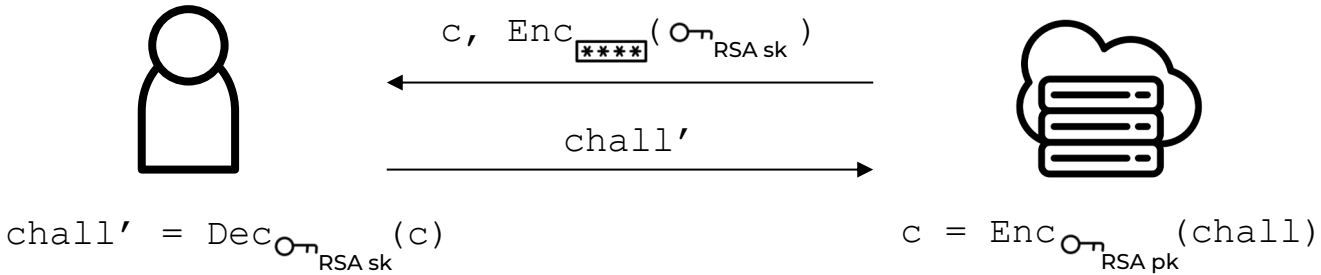


*strongly simplified

Attack 1: RSA auth key recovery

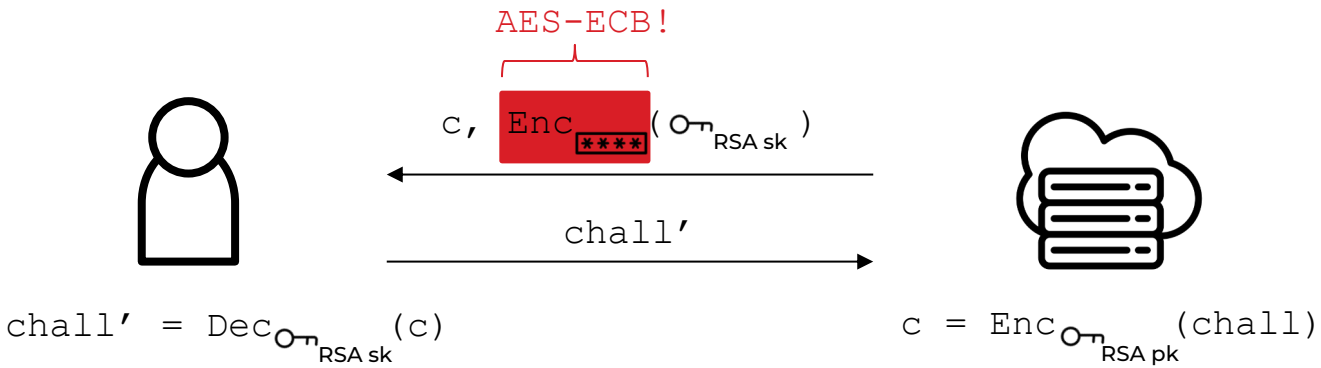
Attack 1 – use of AES-ECB

Exploiting the non-authenticated encryption in the authentication protocol.



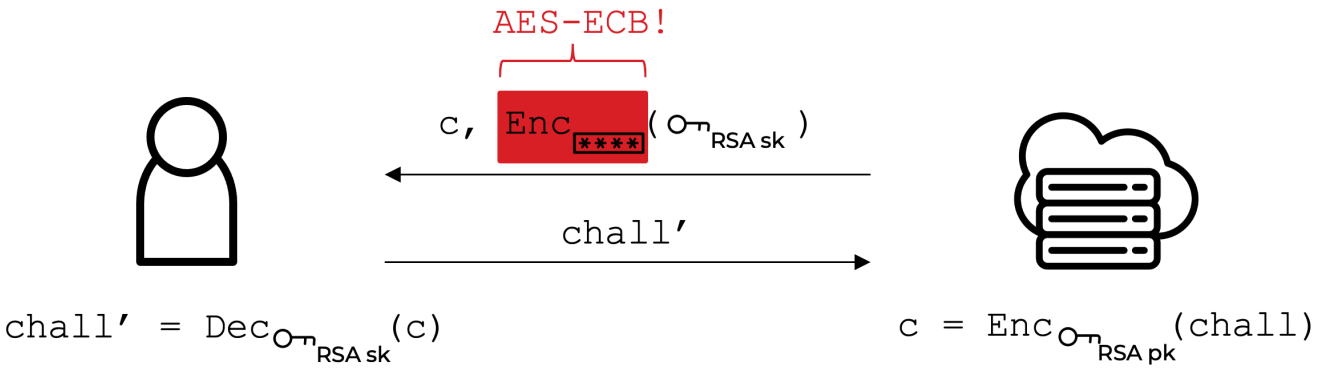
Attack 1 – use of AES-ECB

Exploiting the non-authenticated encryption in the authentication protocol.

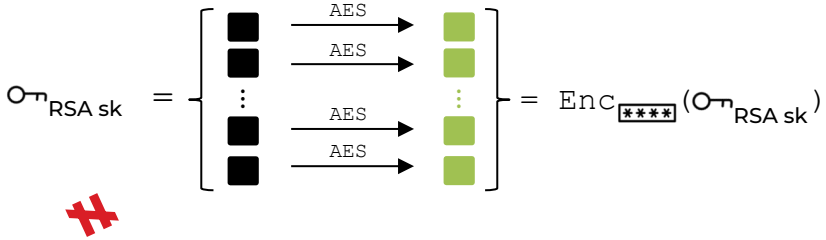


Attack 1 – use of AES-ECB

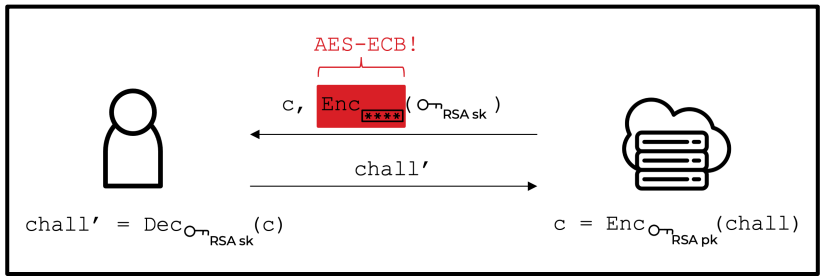
Exploiting the non-authenticated encryption in the authentication protocol.

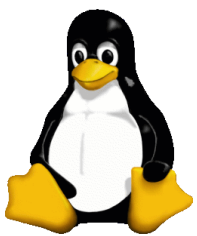


Attack 1 – use of AES-ECB

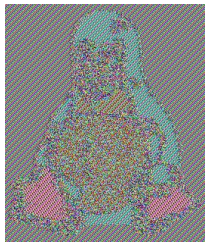


authentication protocol*





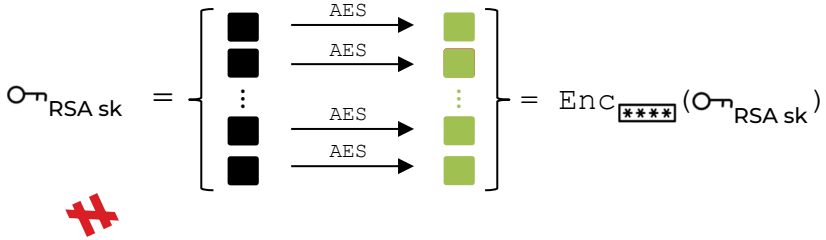
AES-ECB



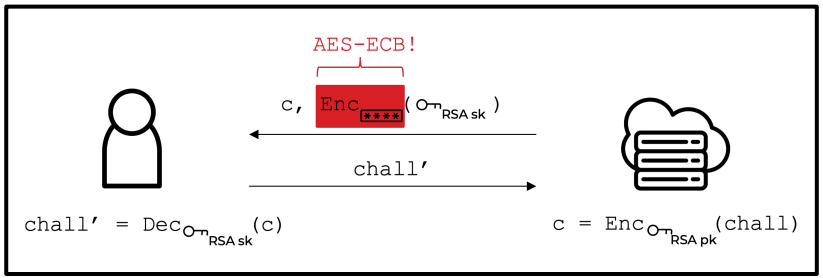
The problem is **not** that AES-ECB is deterministic!

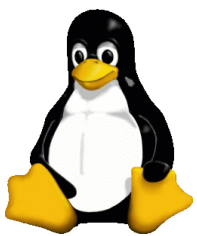
Credit for ECB penguin: Anthony Biondo. https://tonybox.net/posts/ecb-penguin/. (Visited 05/2023).

Attack 1 – use of AES-ECB

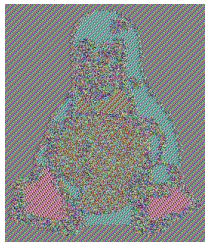


authentication protocol*





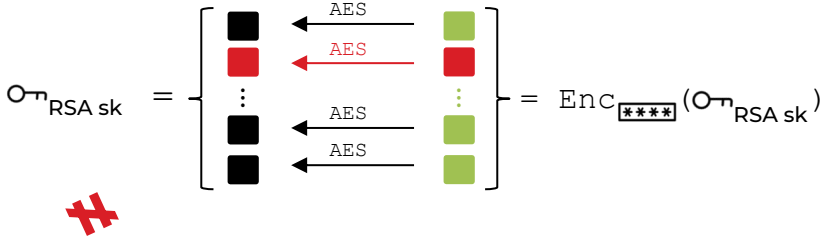
AES-ECB



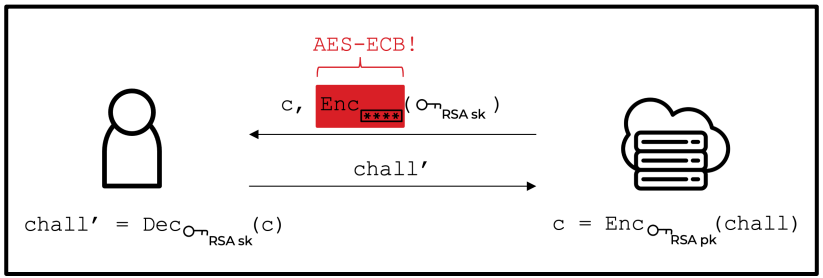
The problem is that AES-ECB is **malleable**!

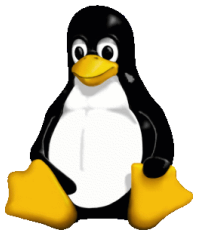
Credit for ECB penguin: Anthony Biondo. https://tonybox.net/posts/ecb-penguin/. (Visited 05/2023).

Attack 1 – use of AES-ECB

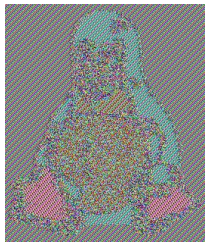


authentication protocol*





AES-ECB

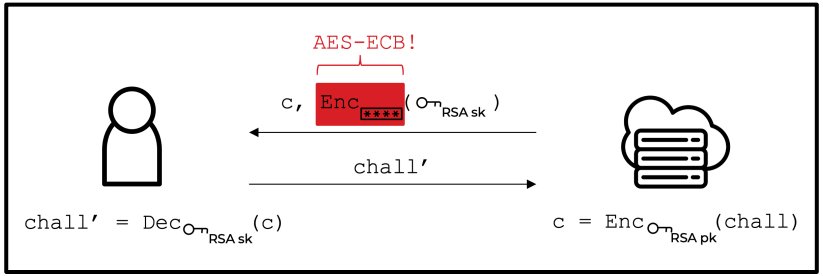


The problem is that AES-ECB is **malleable**!

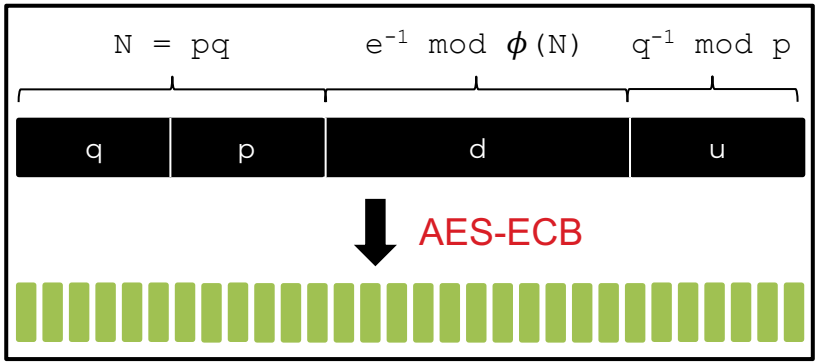
Credit for ECB penguin: Anthony Biondo. https://tonybox.net/posts/ecb-penguin/. (Visited 05/2023).

Attack 1 – RSA key format*

authentication protocol*



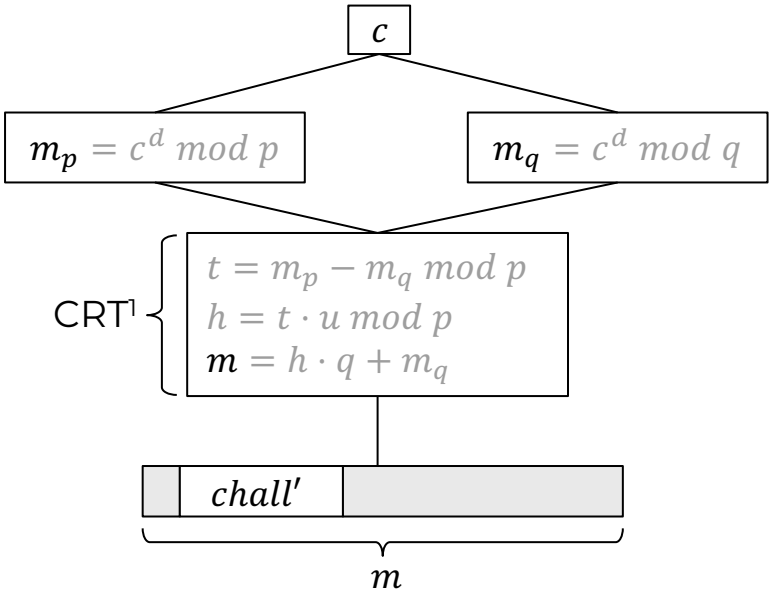
RSA key format*



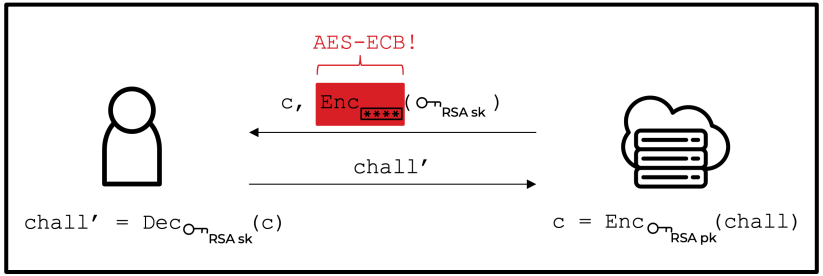
*simplified

Attack 1 – RSA-CRT decryption

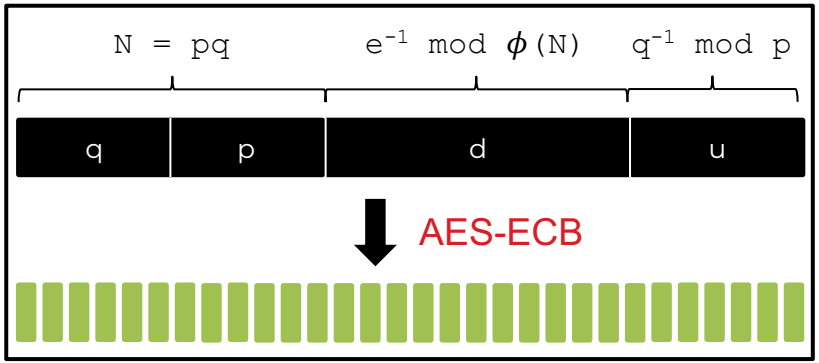
Decrypt in $\mathbb{Z}_p$ and $\mathbb{Z}_q$ reconstruct $m \in \mathbb{Z}_N$.



authentication protocol*



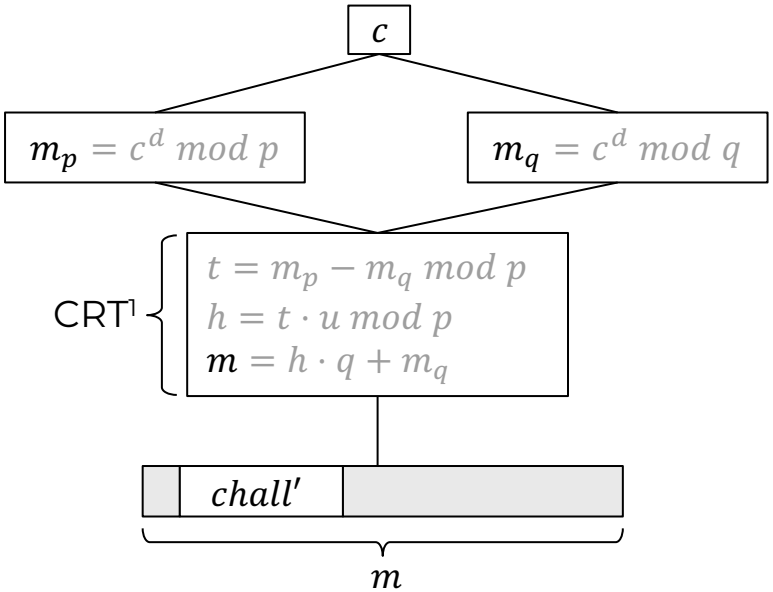
RSA key format*



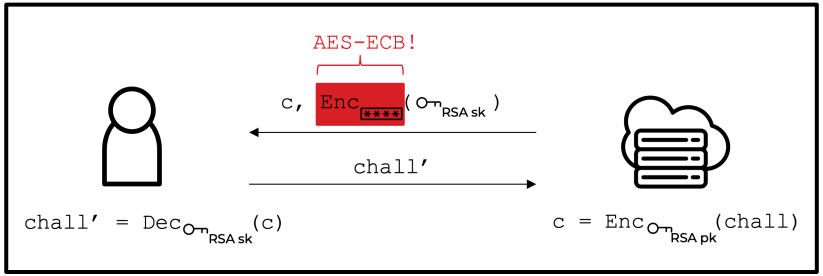
*simplified, [1]Chinese Remainder Theorem

Attack 1 – key ciphertext tampering

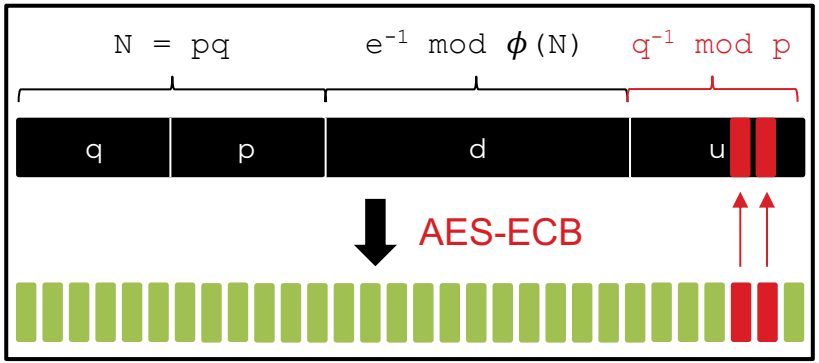
Tampering with u



authentication protocol*



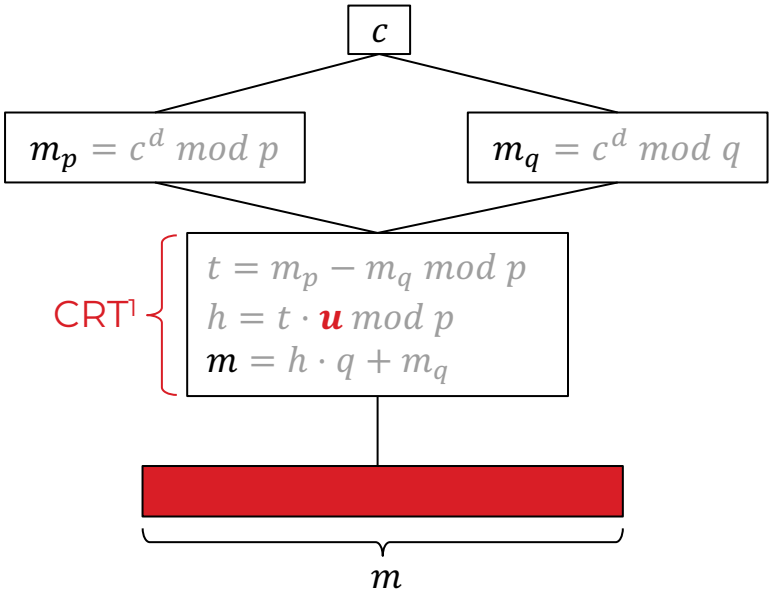
RSA key format*



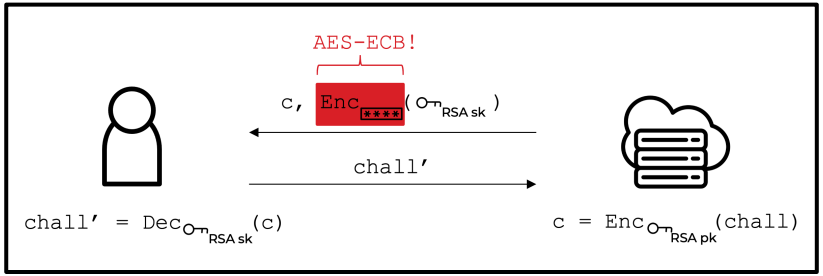
*simplified, [1]Chinese Remainder Theorem

Attack 1 – key ciphertext tampering

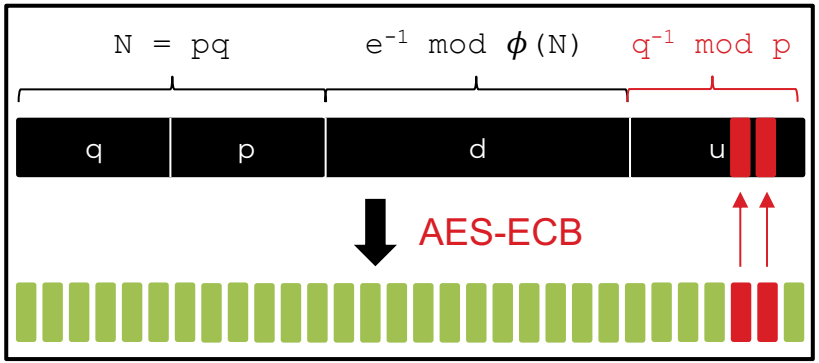
Tampering with u invalidates decryption.



authentication protocol*



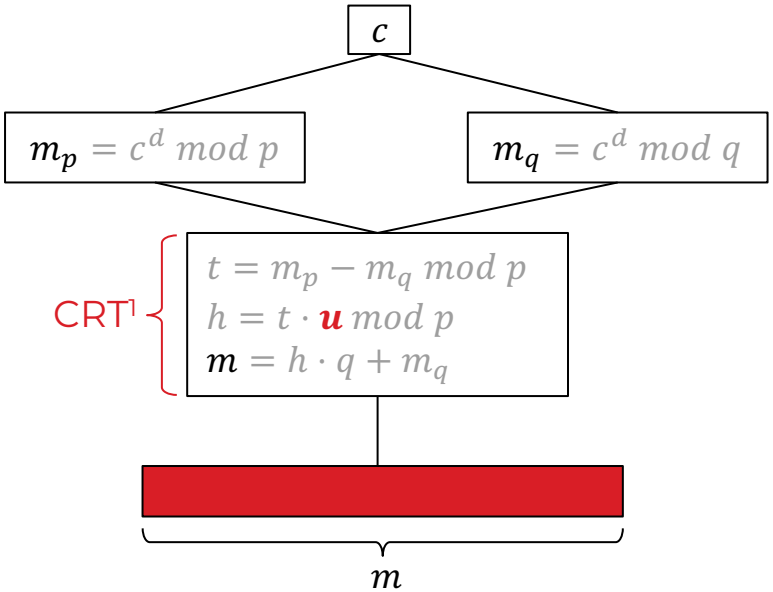
RSA key format*



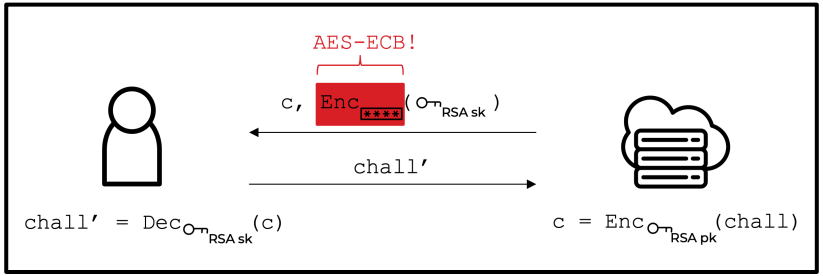
*simplified, [1]Chinese Remainder Theorem

Attack 1 – key ciphertext tampering

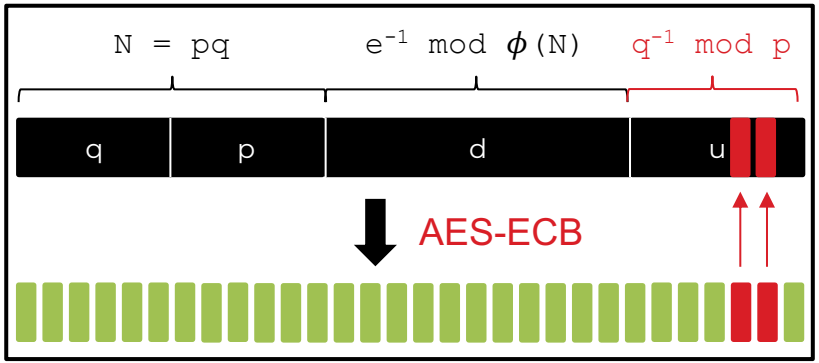
But decryption still succeeds for $chall < p, q$.



authentication protocol*



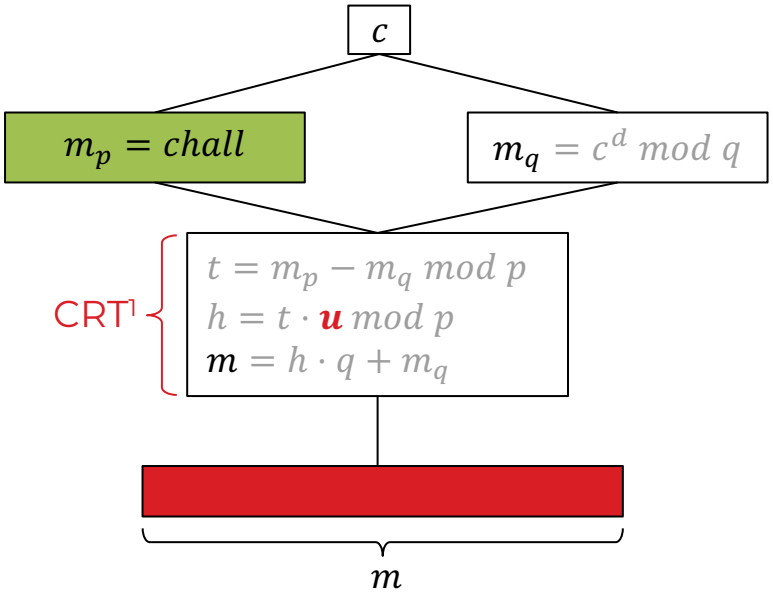
RSA key format*



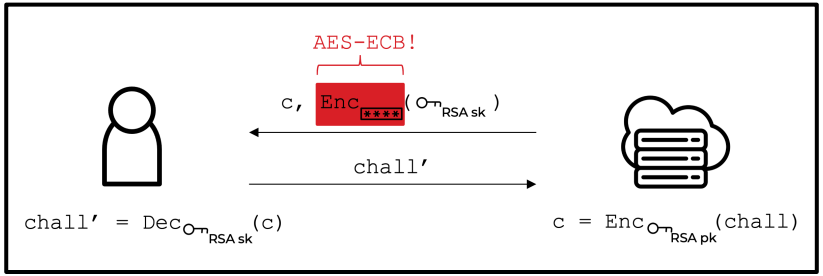
*simplified, [1]Chinese Remainder Theorem

Attack 1 – key ciphertext tampering

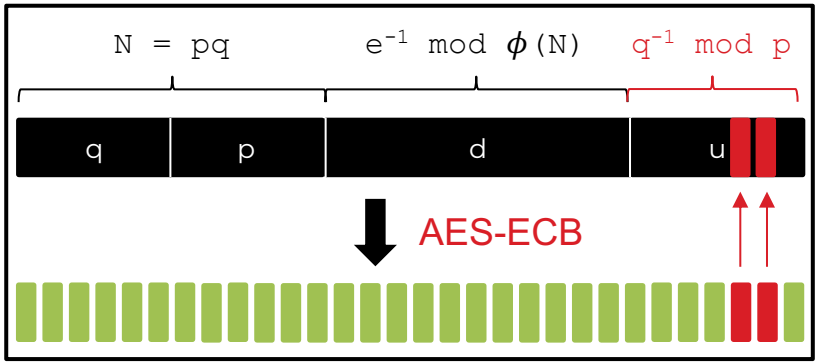
But decryption still succeeds for $chall < p, q$.



authentication protocol*



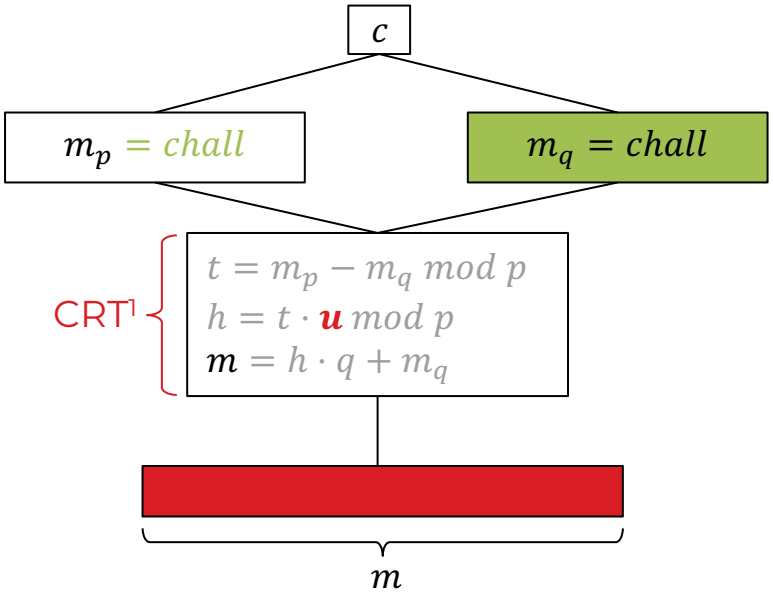
RSA key format*



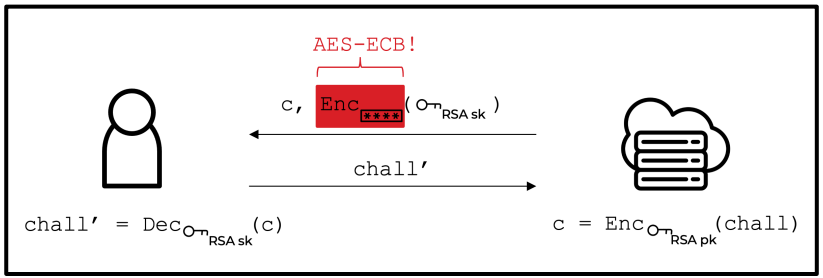
*simplified, [1]Chinese Remainder Theorem

Attack 1 – key ciphertext tampering

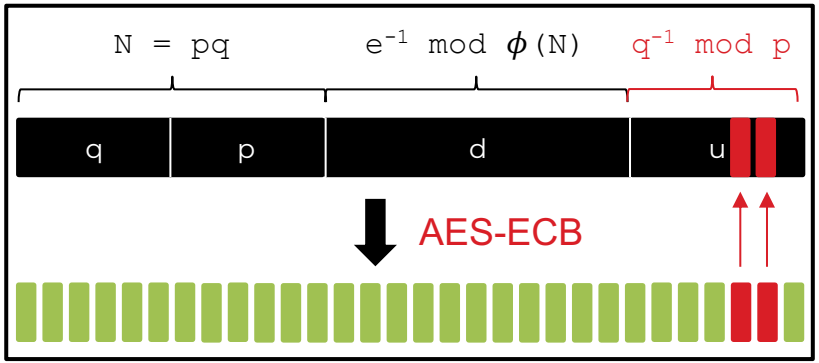
But decryption still succeeds for $chall < p, q$.



authentication protocol*



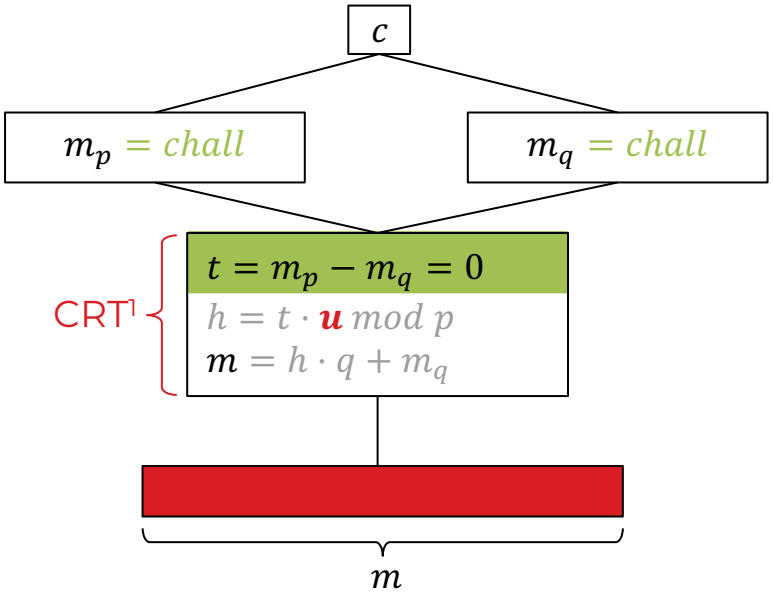
RSA key format*



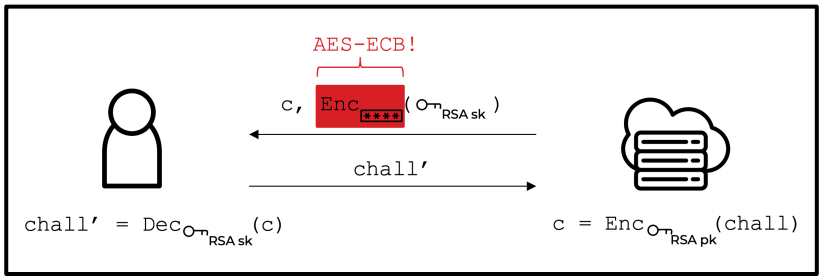
*simplified, [1]Chinese Remainder Theorem

Attack 1 – key ciphertext tampering

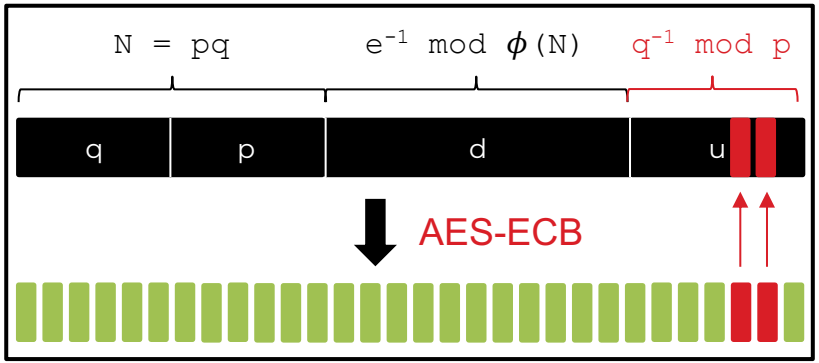
But decryption still succeeds for $chall < p, q$.



authentication protocol*



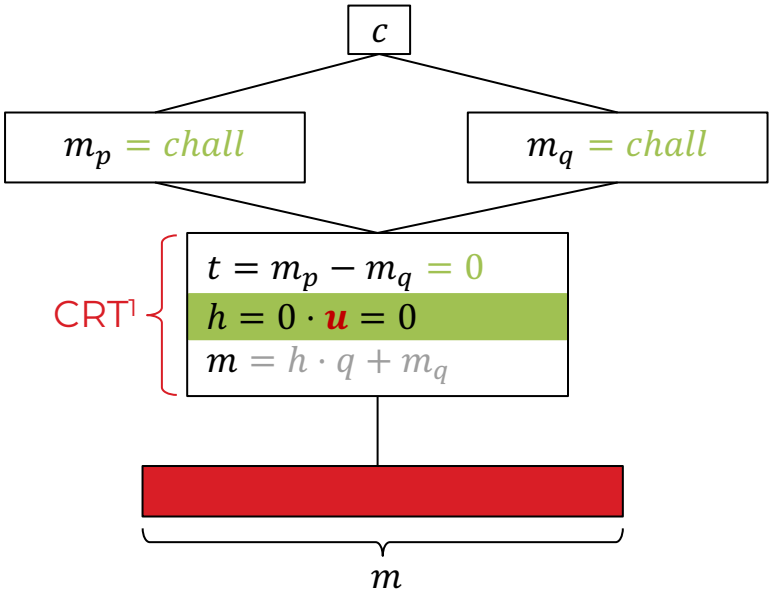
RSA key format*



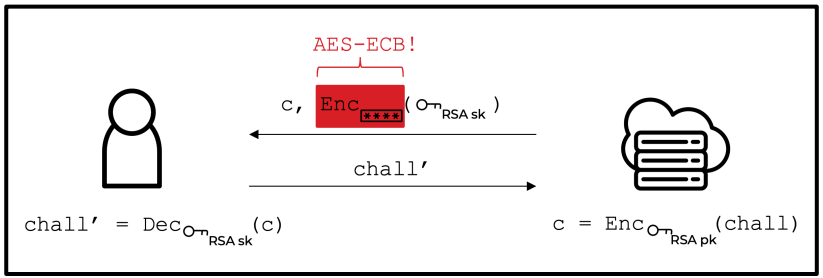
*simplified, [1]Chinese Remainder Theorem

Attack 1 – key ciphertext tampering

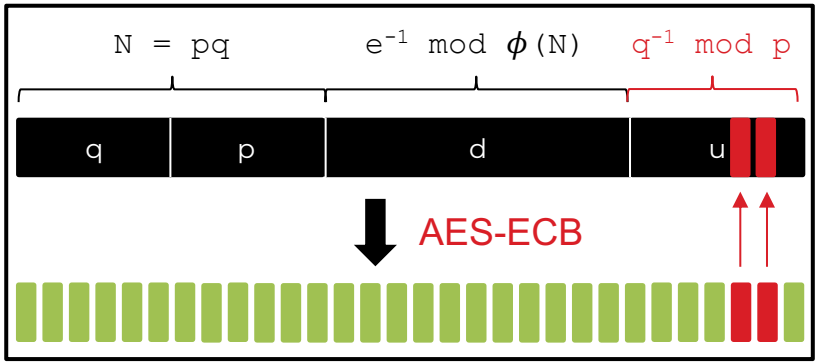
But decryption still succeeds for $chall < p, q$.



authentication protocol*



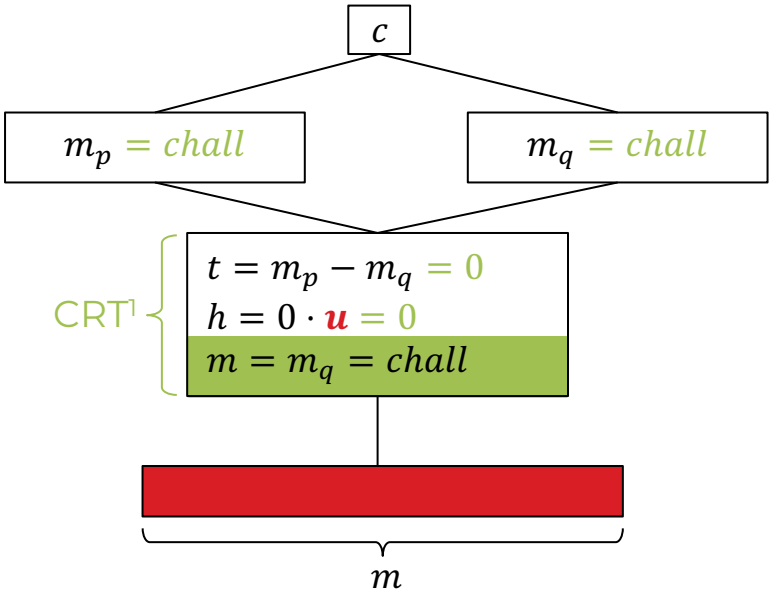
RSA key format*



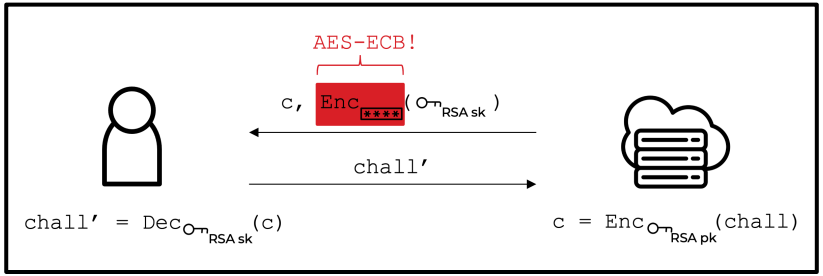
*simplified, [1]Chinese Remainder Theorem

Attack 1 – key ciphertext tampering

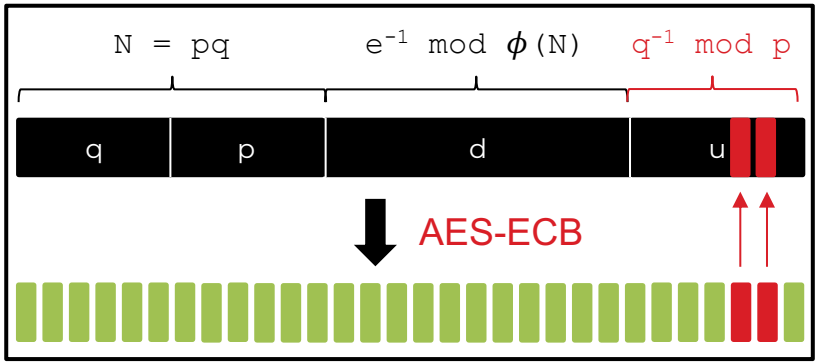
But decryption still succeeds for $chall < p, q$.



authentication protocol*



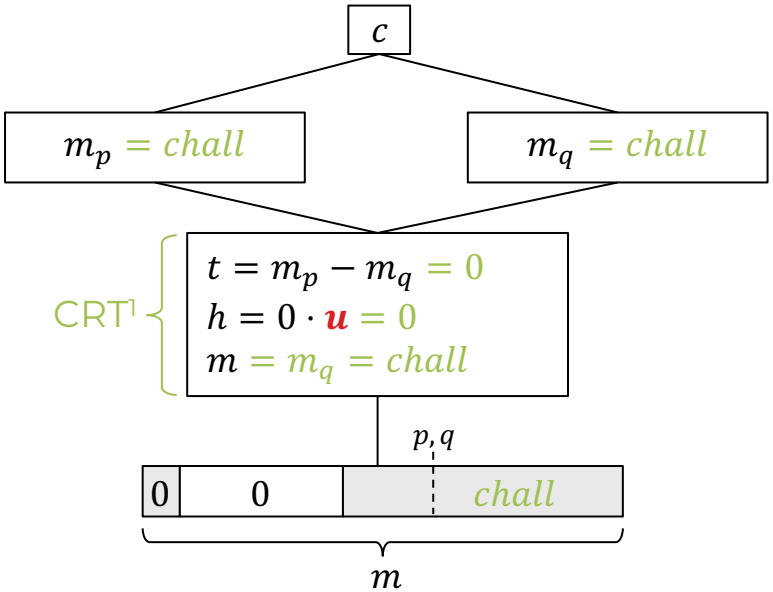
RSA key format*



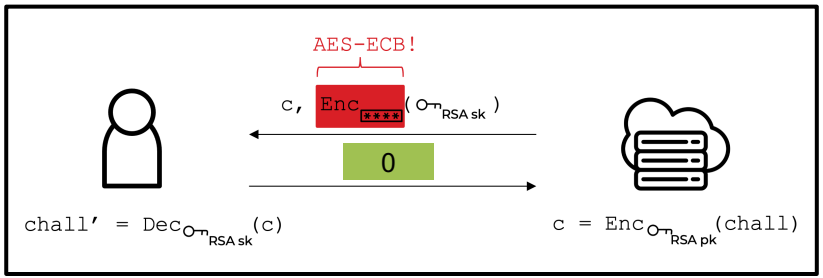
*simplified, [1]Chinese Remainder Theorem

Attack 1 – key ciphertext tampering

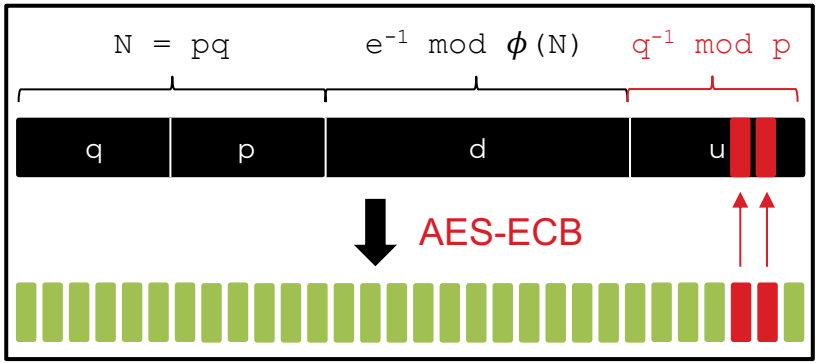
But decryption still succeeds for $chall < p, q$.



authentication protocol*



RSA key format*

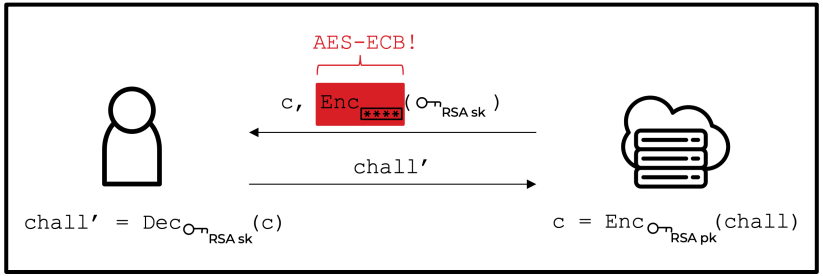


*simplified, [1]Chinese Remainder Theorem

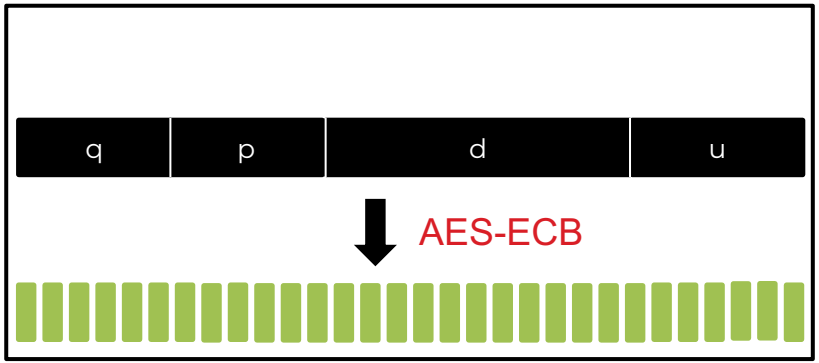
Attack 1 – summary

Binary search for primes p, q.

authentication protocol*



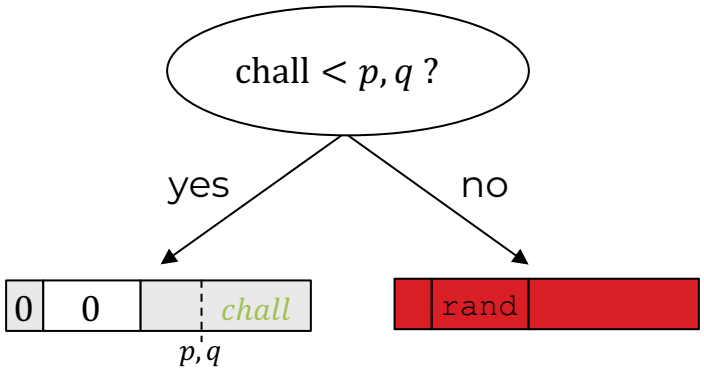
RSA key format*



*simplified

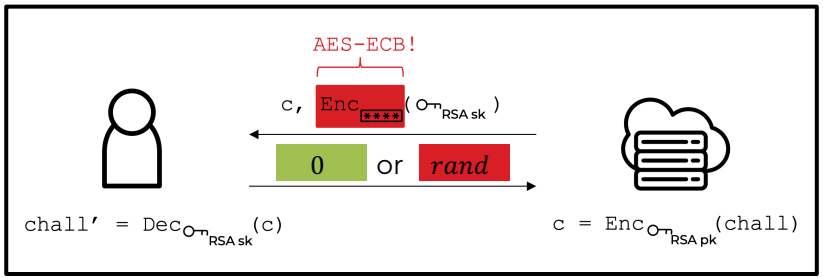
Attack 1 – summary

Binary search for primes p, q.

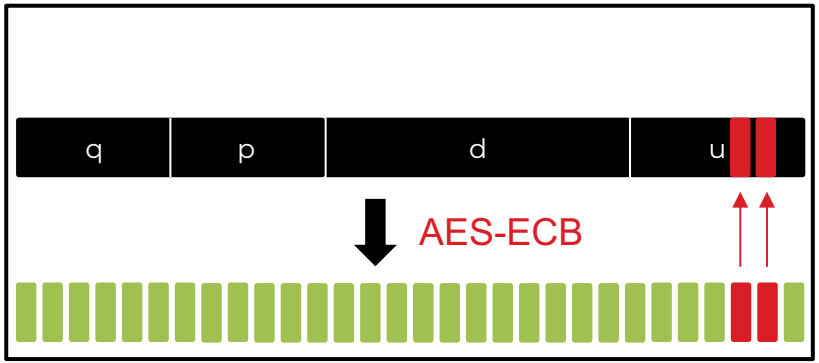


Recover RSA sk in 512 login attempts
(later improved by [RH23] and [AHMP23])

authentication protocol*



RSA key format*



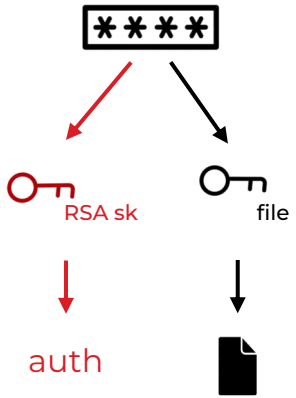
*simplified

[RH23] Ryan, Keegan, and Heninger, Nadia. "The Hidden Number Problem with Small Unknown Multipliers: Cryptanalyzing MEGA in Six Queries and Other Applications." Public-Key Cryptography. 2023.

[AHMP23] Albrecht, Martin, Haller, Miro, Mareková, Lenka, Paterson, Kenny. "Caveat Implementor! Key Recovery Attacks on MEGA." Eurocrypt. 2023.

Attack 1 – impact

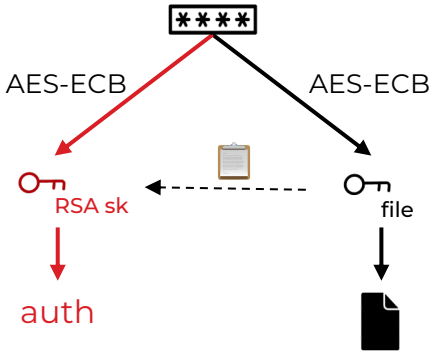
- Compromised: RSA secret key (not: files)



Attack 2: file decryption

Attack 2 – AES-ECB, again!

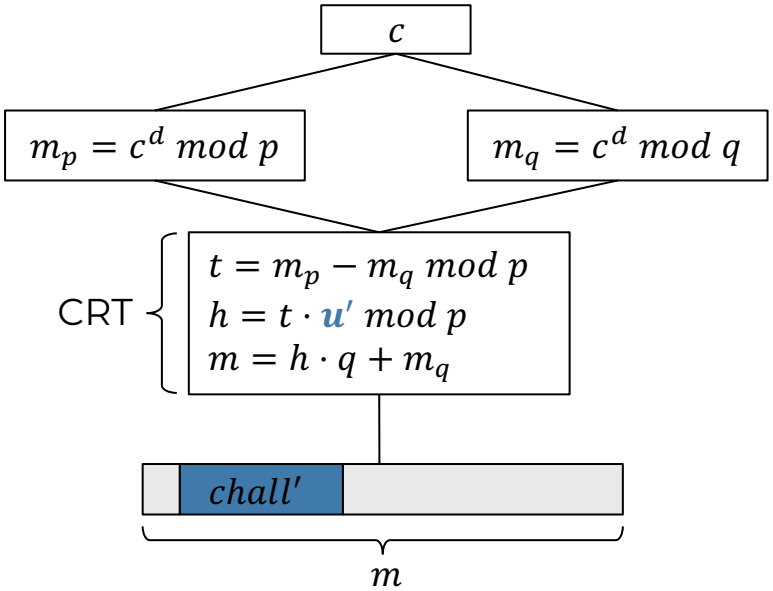
- File keys are also encrypted with AES-ECB!
- Idea:
  - Cut and paste file key ciphertext blocks into the RSA secret key ciphertext
  - Target authentication protocol again



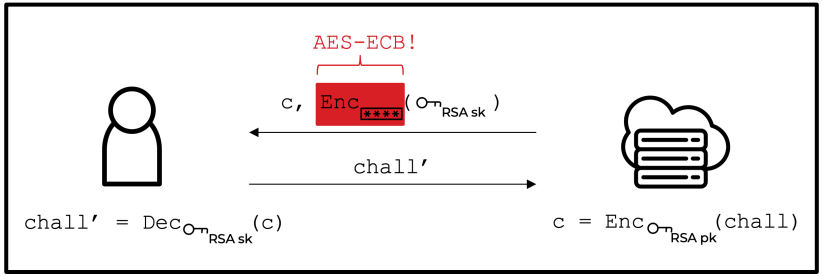
*simplified

Attack 2 – simplifying RSA-CRT

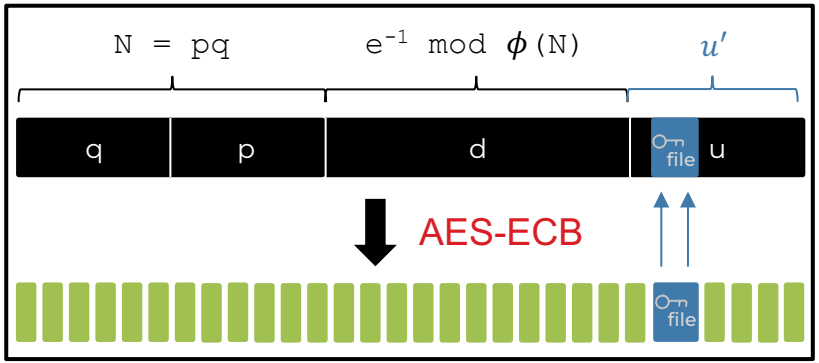
Pick chall to simplify RSA-CRT equations, recover file key from u'.



authentication protocol*



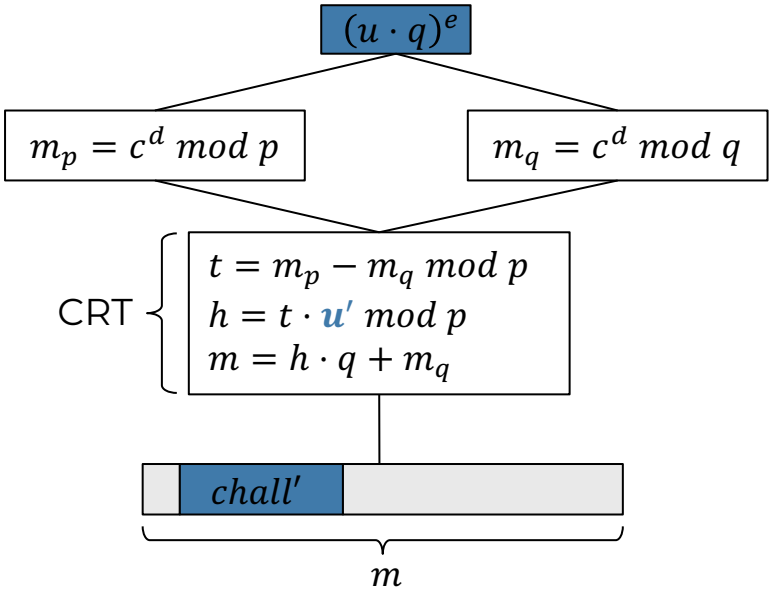
RSA key format*



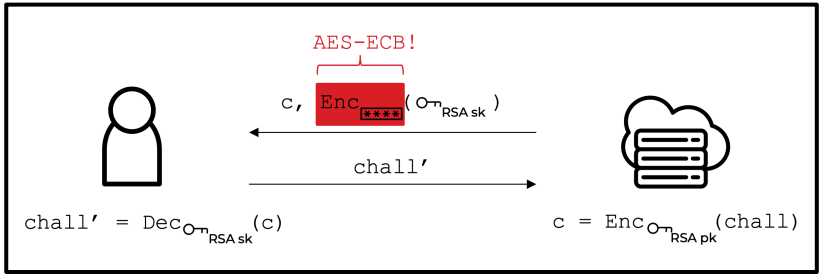
*simplified

Attack 2 – simplifying RSA-CRT

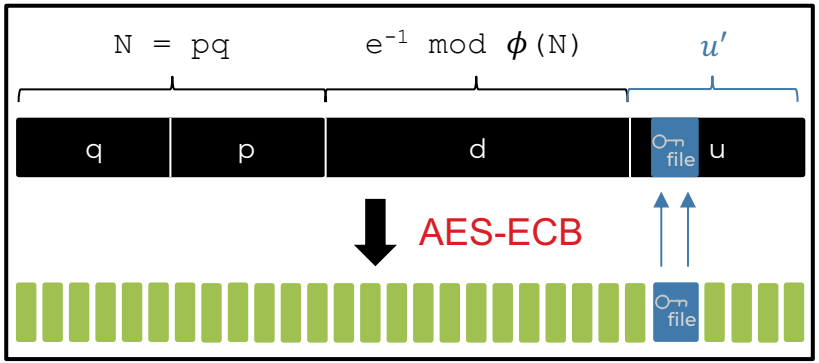
Pick chall to simplify RSA-CRT equations, recover file key from u'.



authentication protocol*



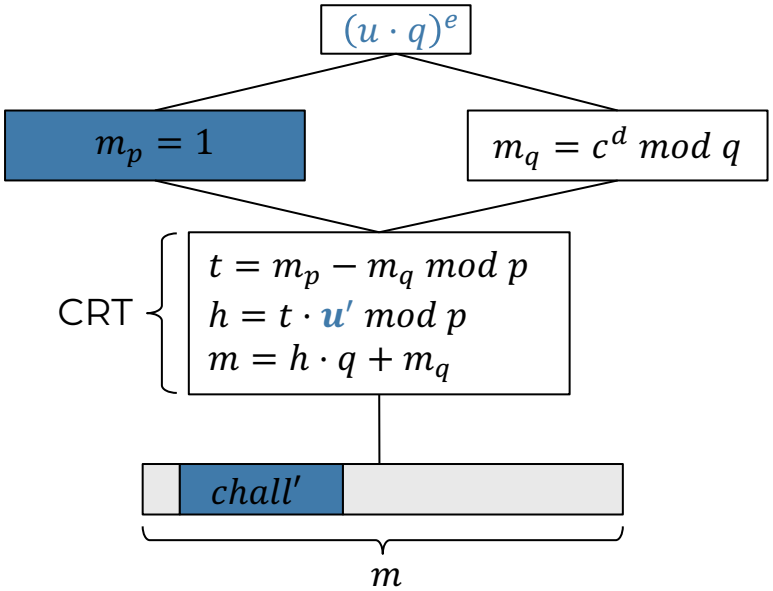
RSA key format*



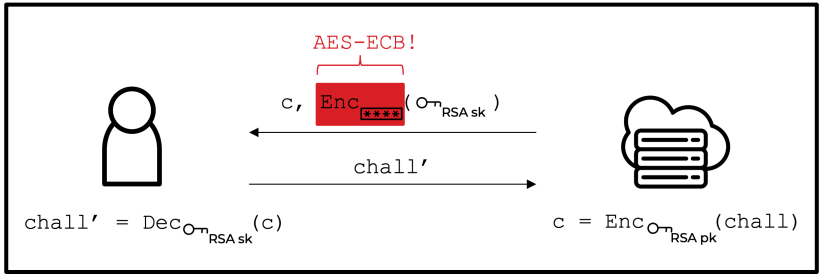
*simplified

Attack 2 – simplifying RSA-CRT

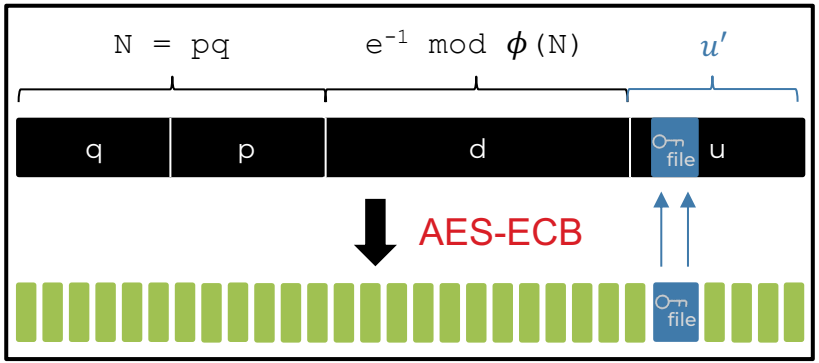
Pick chall to simplify RSA-CRT equations, recover file key from u'.



authentication protocol*



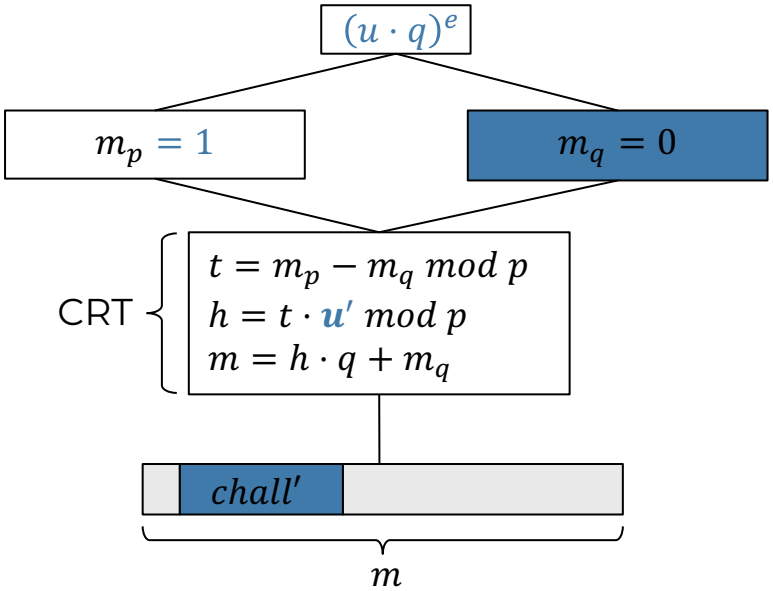
RSA key format*



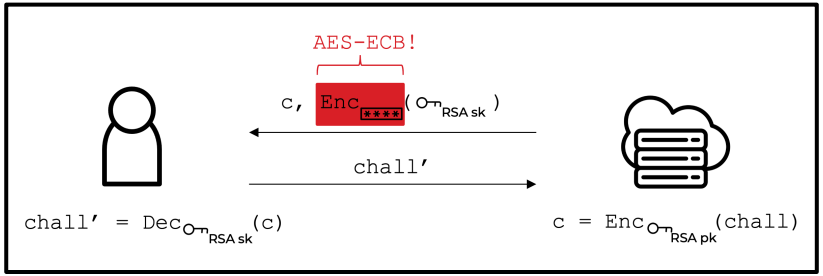
*simplified

Attack 2 – simplifying RSA-CRT

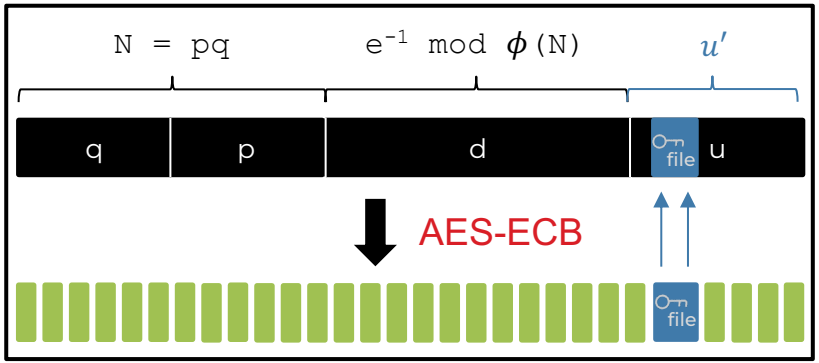
Pick chall to simplify RSA-CRT equations, recover file key from u'.



authentication protocol*



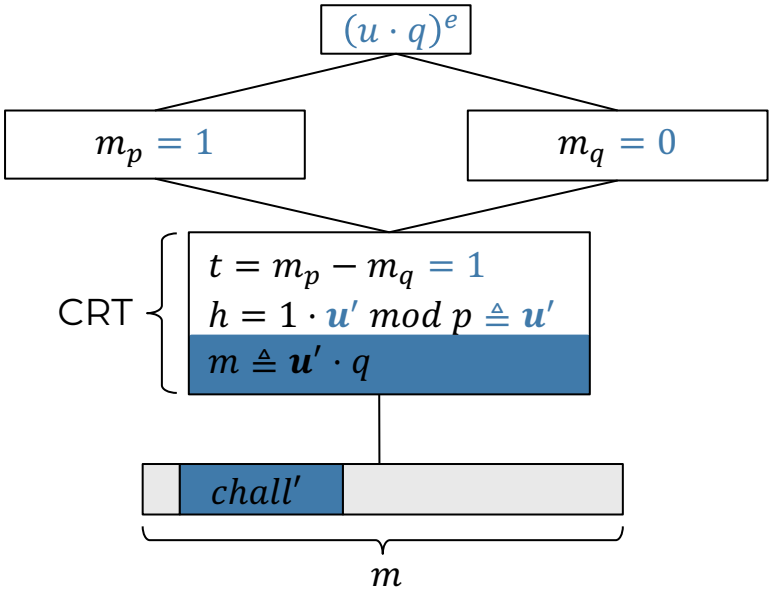
RSA key format*



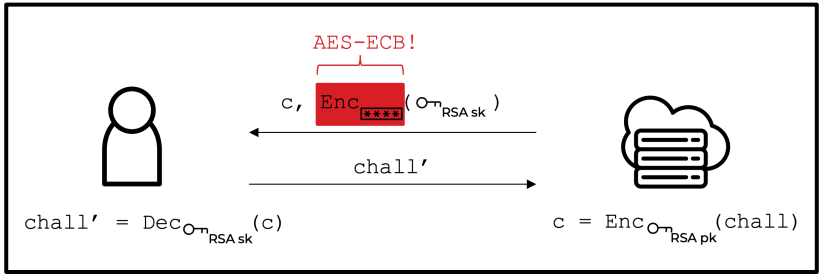
*simplified

Attack 2 – simplifying RSA-CRT

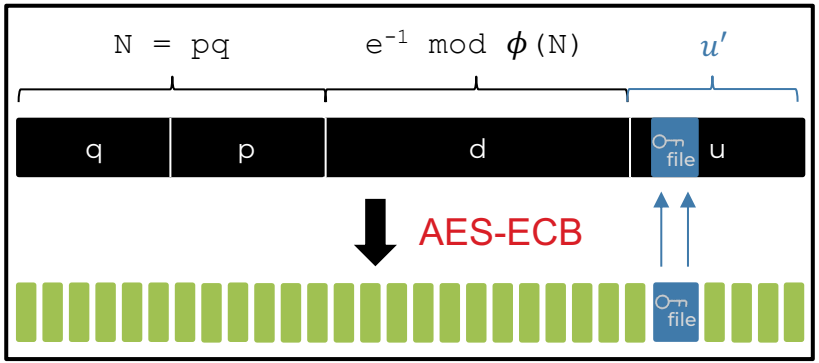
Pick chall to simplify RSA-CRT equations, recover file key from u'.



authentication protocol*



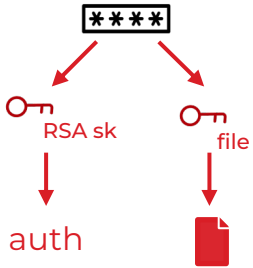
RSA key format*



*simplified, ^with high probability

Attack 2 – summary and impact

- Attack 2:
  - Cut and paste file key ciphertexts into RSA sk
  - Decrypt one file key per login attempt
- Compromises confidentiality of all user files

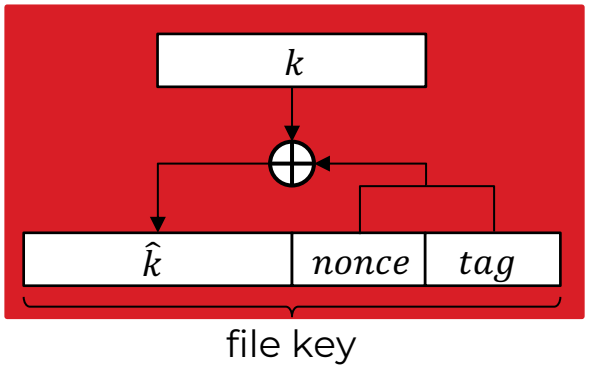


Attacks 3 & 4: integrity

Attack 3 & 4 – AES-CCM file encryption*

- AES-CCM:
  - $tag = \mathrm{CBC{-}MAC}(\mathrm{k}, \mathrm{nonce}, \blacksquare)$
  - $ctxt = \mathrm{AES{-}CTR}(\mathrm{k}, \mathrm{nonce}, \blacksquare)$
- File key:
  - XOR of AES key, nonce, and MAC tag

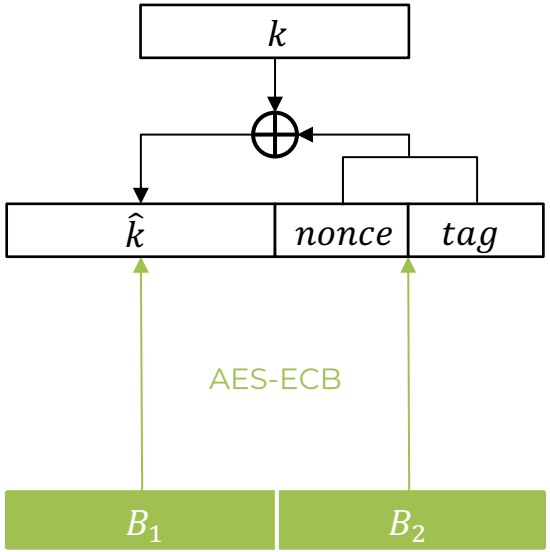
strange construction!



*very simplified

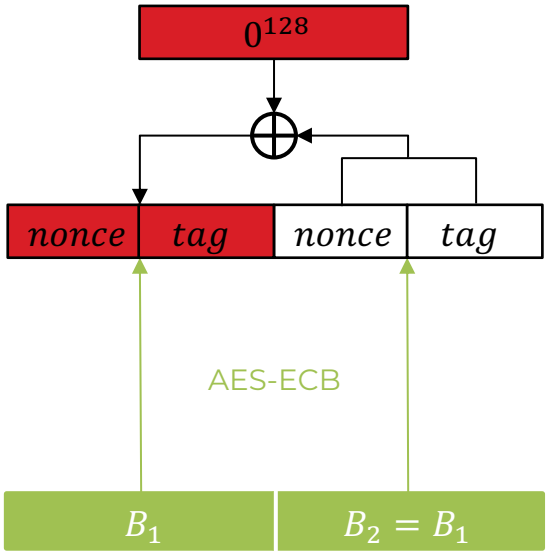
Attack 3 – still AES-ECB

- File keys encrypted with AES-ECB



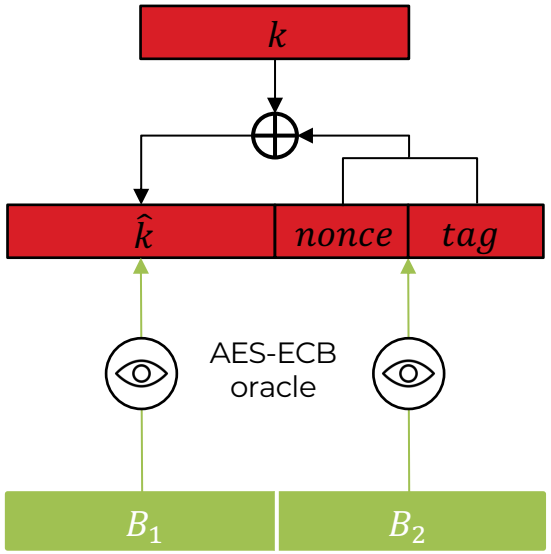
Attack 3 – repeating CT blocks

- File keys encrypted with AES-ECB
- Attack 3:
  - Same ciphertext blocks → key 0^{128}
  - 1 PTXT-CTXT pair to pass authentication
- All-zero key is suspicious



Attack 4 – avoiding detection

- Use attack 2
- Get random key and nonce
- Not detectable



Summary: 5 attacks

Attacks

- Attack 1: RSA key recovery
  - Malleable secret key + oracle
- Attack 2: file key recovery
  - Cut and paste AES ctxt blocks
- Attack 3: integrity attack
  - File forgery under the “zero key”
- Attack 4: framing attack
  - Like attack 3, but not detectable
- Attack 5: Bleichenbacher
  - Adapted to MEGA’s RSA padding

Root causes

- No AE for key encryption
- Missing key separation
- Rolling your own crypto
- Cryptographic agility
- Backwards compatibility

Towards secure cloud storage

Cloud Storage Standard

- Standardization effort...
  - ...involving various stakeholders
  - ...to design a well-analysed and practical E2EE cloud storage system

Thank you!
Questions?

Paper: "**MEGA**: Malleable Encryption Goes Awry"

Website: mega-awry.io

Attacks PoC: github.com/MEGA-Awry

Additional references:

Icons from the Noun Project by: arif fauzi hakim, M Yudi Maulana, alrigel, Oh Rian, rukanicon, Тимур Минвалеев, Ami Ho, juli, Andrew Doane, Eucalyp, Symbolon, Adrien Coquet, Rediffusion, sahara junadir