Key Overwriting Attacks

Miro Haller

CSE 207b; Nov 30, 2023

Lecture Resources

- Kenneth G. Paterson. ”On The Unreasonable Effectiveness of Key Overwriting”. WAC 6. August 20, 2023. (link)
- Lara Bruseghini, Daniel Huigens, Kenneth G. Paterson. “Victory by KO: Attacking OpenPGP Using Key Overwriting”. CCS 2022. (link)
- Matilda Backendal, Miro Haller and Kenneth G. Paterson. “MEGA: Malleable Encryption Goes Awry”. IEEE S&P 2023. (link)
- Martin R. Albrecht, Miro Haller, Lenka Mareková, Kenneth G. Paterson. “*Caveat Implementor!* Key Recovery Attacks on MEGA”. Eurocrypt 2023. (link)

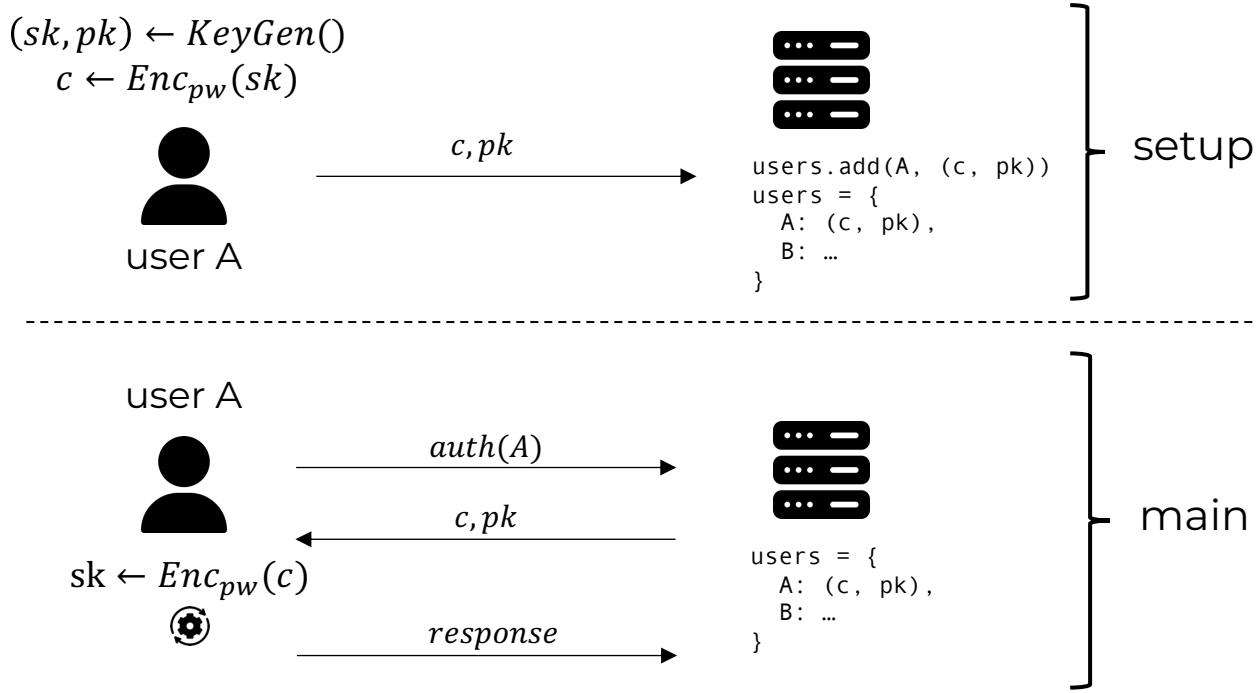
KO Attacks Overview

KO Attacks: In Context

UNKEYED	PRG → Low entropy seed, Dual EC DRBG, bias in RC4 in TLS Hash functions → Length extension attacks
SYMMETRIC	Stream ciphers → Key stream reuse Block ciphers → CBC padding oracle
ASYMMETRIC	Key exchange: (EC)DH Encryption: RSA → Bleichenbacher's attack on PKCS#1v1.5, shared factors ECC → Invalid curve attack Signature: (EC)DSA → Nonce reuse
TOOLS	DLOG → Pohlig-Hellman, BSGS, Pollard-Rho Factoring → NFS, index calculus Lattices → Coppersmith's method **Key overwriting attacks** → OpenPGP, MEGA (protocol level)

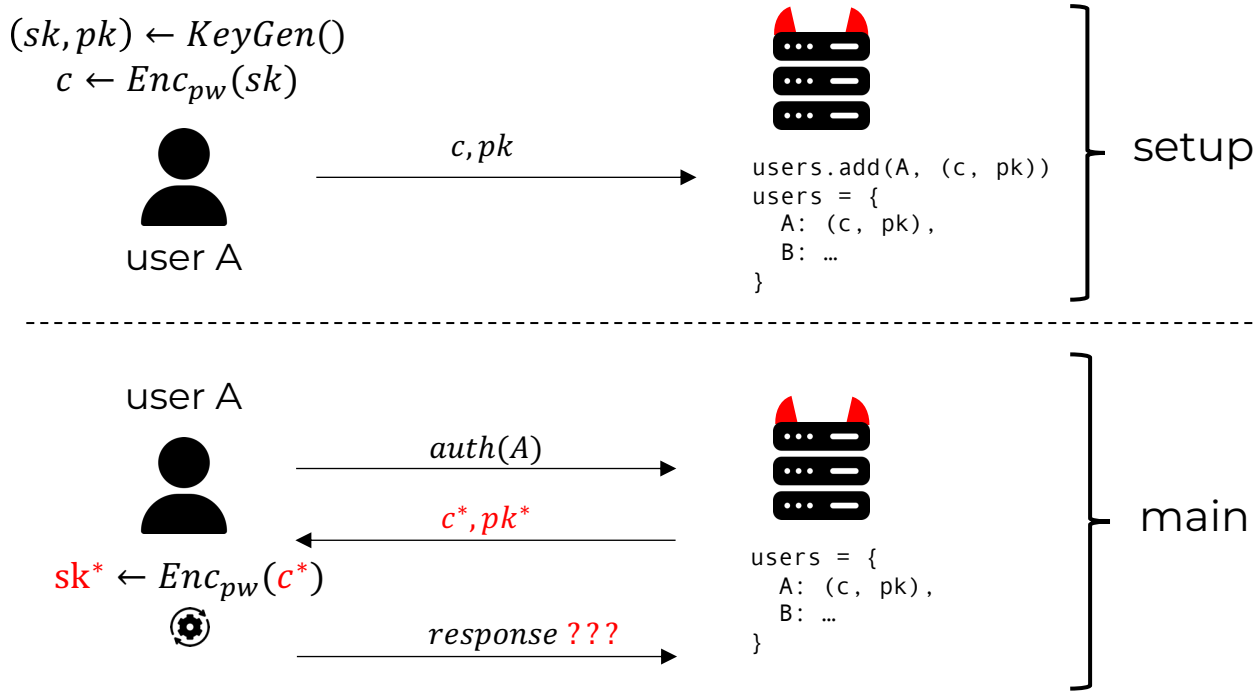
KO Attacks: Setting

- Interactive protocol
- Setup phase:
  - Client uploads encrypted secret key
- Main phase:
  - User authenticates
  - Fetches encrypted key material
  - Decrypts keys and performs some operations
  - Eventually responds to server



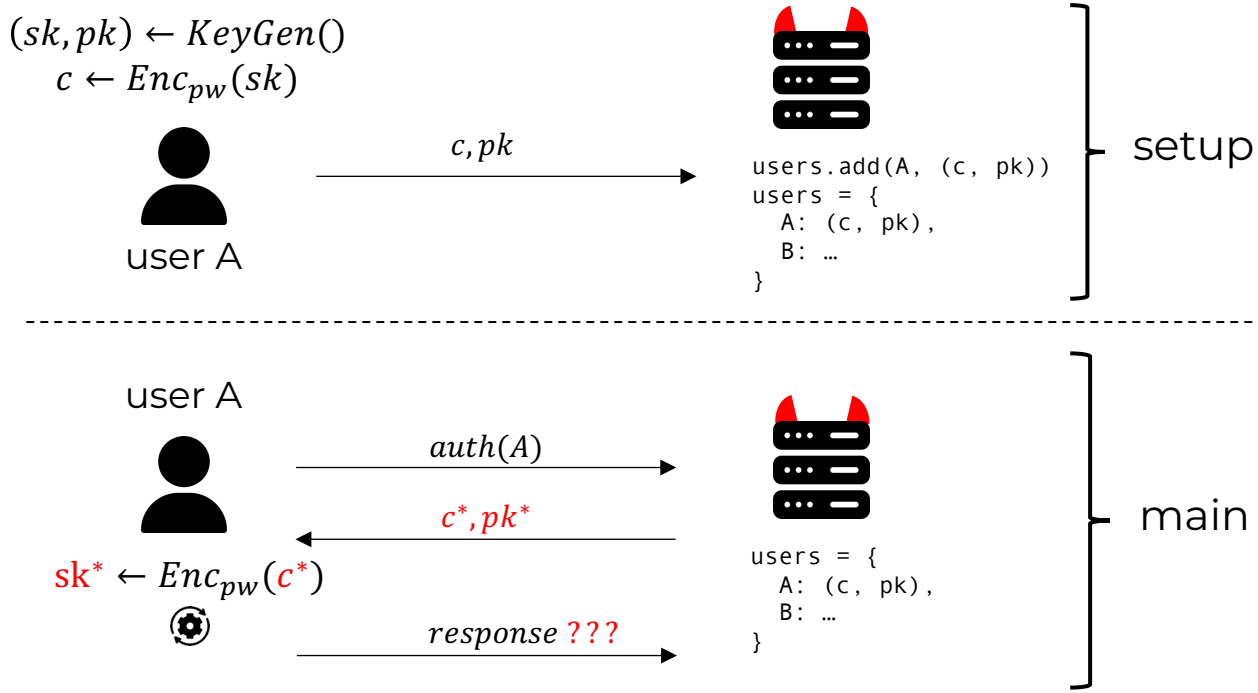
KO Attacks: Threat Model

- Malicious server, e.g.
  - Compromised
  - Compelled to comply
- May overwrite outsourced key material
- Observe computation with bogus key material



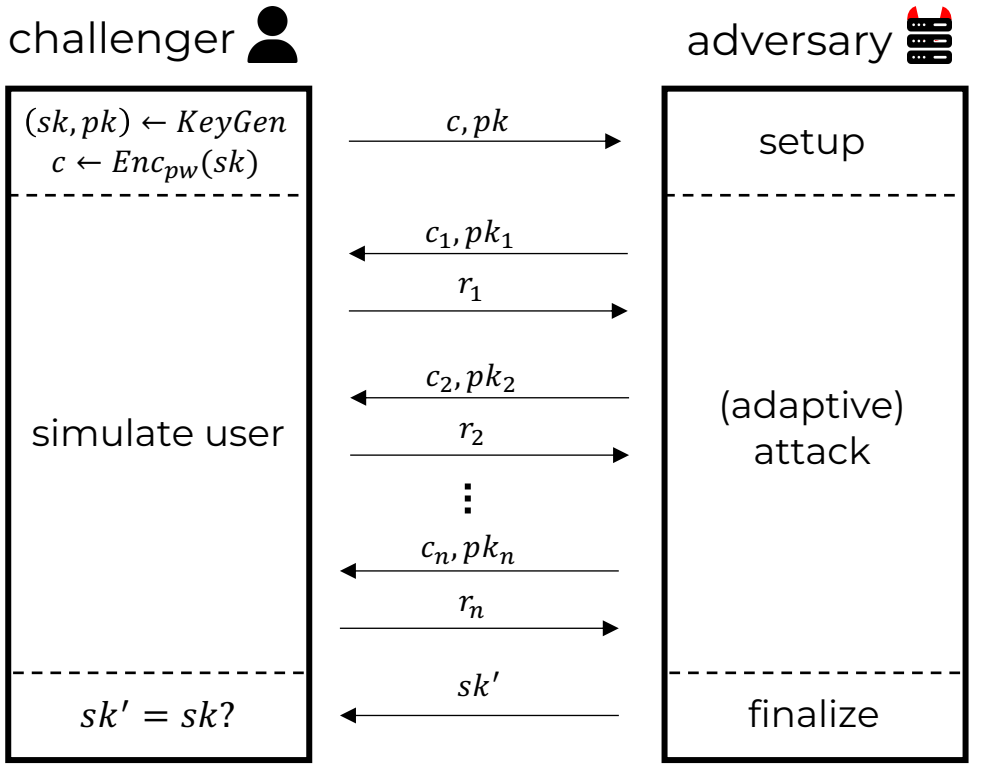
KO Attacks: Goal and Variants

- Attack goal:
  - Key recovery (of sk)
  - (or weaker goals like decryption)
- Overwriting control varies:
  - only pk,
  - both c and pk,
  - or controlled vs. blind overwriting?



KO Key Recovery (KO-KR) Game

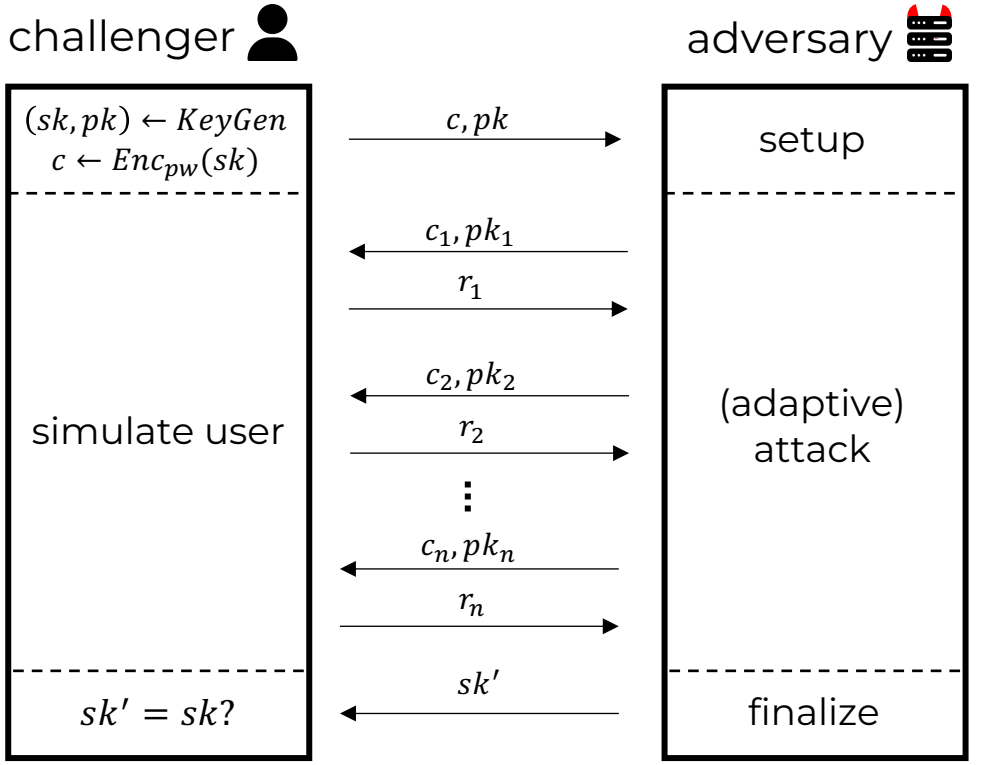
- Adversary receives valid encryption of secret key
- Overwrite user keys n times with (c_i, pk_i) and observe client responses r_i
- Guess secret key sk'
- KO-KR-security:
  - No efficient adversary wins KO-KR game with non-negl. success probability



KO Key Recovery (KO-KR) Game

Question:
IND-CPA security of $Enc \Longrightarrow$ KO-KR secure?

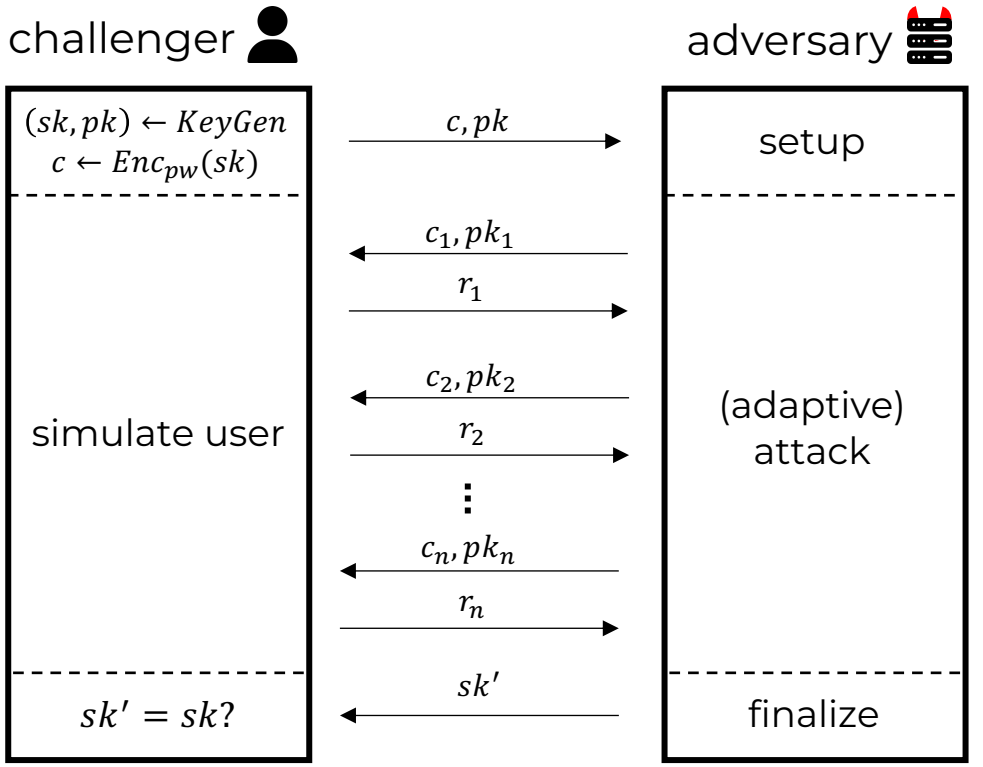
- KO-KR-security:
  - No efficient adversary wins KO-KR game with non-negl. success probability



KO Key Recovery (KO-KR) Game

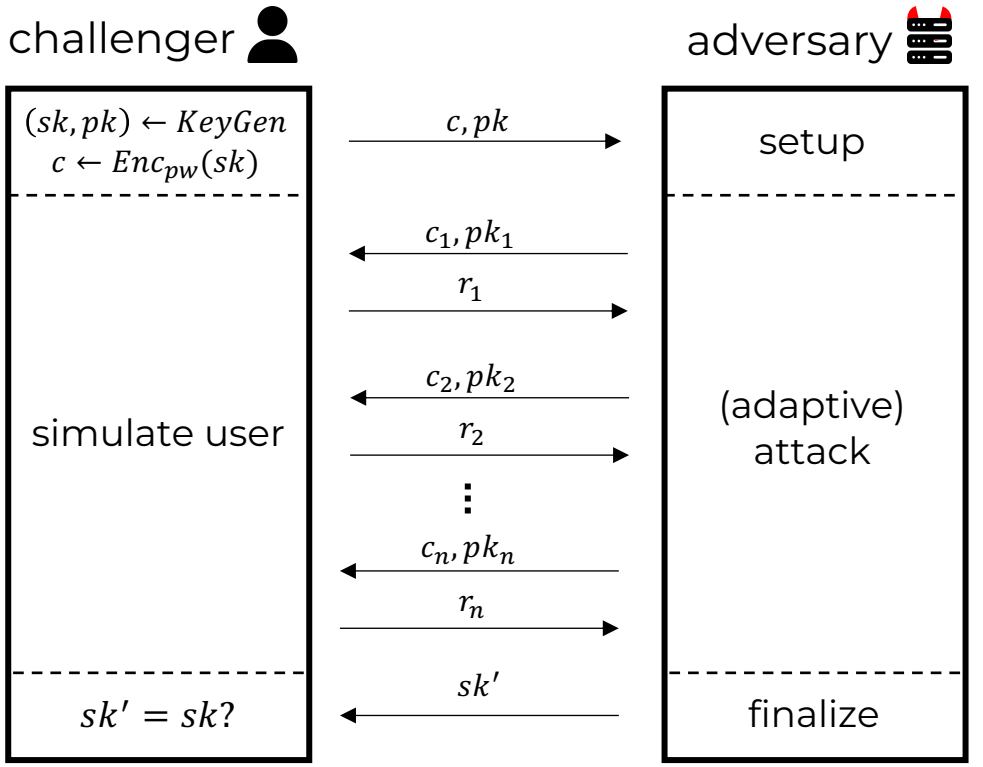
Question:
IND-CPA security of $Enc \Longrightarrow$ KO-KR secure?

Answer:
No, the IND-CPA adversary can't simulate users for KO-KR (needs decryption)



KO Key Recovery (KO-KR) Game

- KO-KR-security:
  - No efficient adversary wins KO-KR game with non-negl. success probability
- Efficient in practice:
  - Minimize n
  - Minimize require user interaction (may be necessary to produce response)



Key Overwriting Attacks

- Questions?
  - On the setting (setup & main phases)
  - On the thread model (malicious server)
  - On the security definition?

- Up next: KO attack on OpenPGP


KO Attack on OpenPGP

OpenPGP

- PGP: Pretty Good Privacy
- Standard (RFC 4880) to secure electronic communication and data
- Used for email encryption, secure storage, and data authentication

Discuss with your neighbors:

- Have you ever used PGP? If yes, how was the experience?
- Is the XKCD on the right a good way to verify email authenticity?

src: https://xkcd.com/1181/ (visited: 11/27/23)

OpenPGP

- PGP: Pretty Good Privacy
- Standard (RFC 4880) to secure electronic communication and data.
- Used for email encryption, secure storage, and data authentication
- Criticized for security and usability
  - Outdated primitives, complex data structures, not provably secure
  - Hard to use (correctly)
- Still used and actively developed

src: https://xkcd.com/1181/ (visited: 11/27/23)

DSA Reminder (cf. Lecture 12)

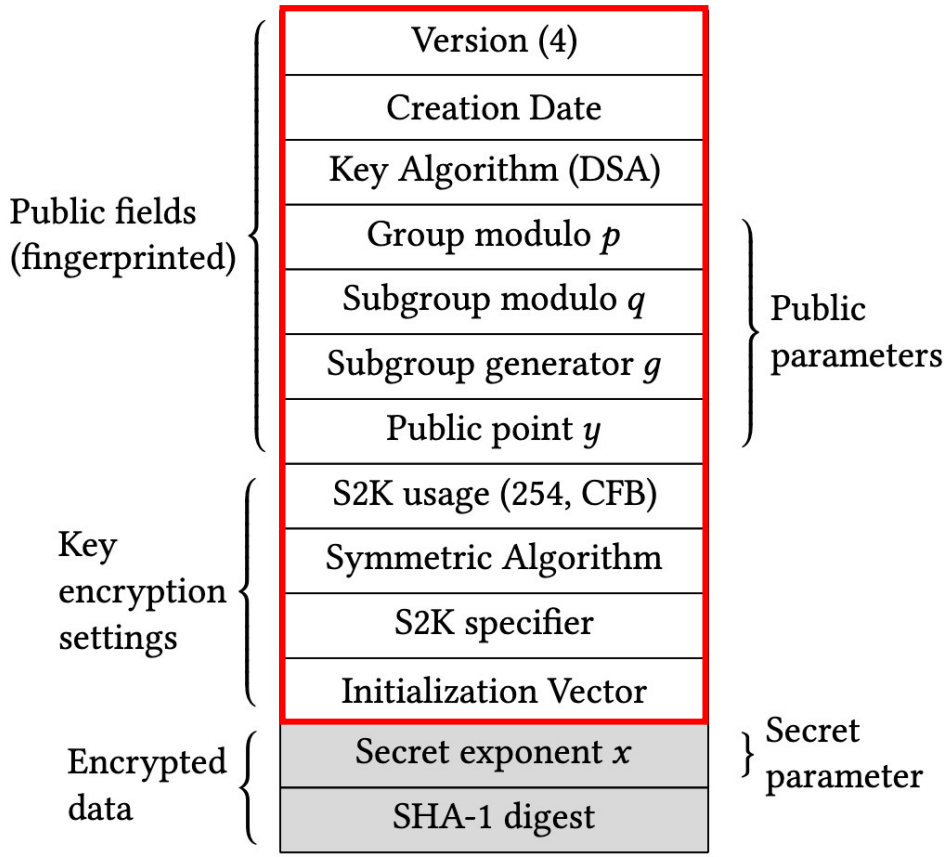
- Secret key: $sk = x \in \mathbb{Z}_q$
- Public parameters:
  - primes p and $q \mid (p-1)$
  - generator g of subgroup with order q
  - hash function $H: \{0,1\}^* \to \mathbb{Z}_q$
  - public key $\mathrm{pk} = (\mathrm{p}, \mathrm{q}, \mathrm{g}, \mathrm{y} = g^x \bmod p)$
- Operations

$Verify(pk = (\mathrm{p}, \mathrm{q}, \mathrm{g}, \mathrm{y}), m, sig = (r, s))$:
$u_1 = H(m)s^{-1} \bmod q$
$u_2 = rs^{-1} \bmod q$
return $r == g^{u_1} y^{u_2} \bmod p \bmod q$

$Sign(sk = x, pk = (\mathrm{p}, \mathrm{q}, \mathrm{g}, \mathrm{y}), m)$:
$k \leftarrow_{\$} \mathbb{Z}_q$ // (sample uniformly random)
$r = g^k \bmod p \bmod q$
$s = k^{-1}(H(m) + xr) \bmod q$
return (r, s)

OpenPGP DSA Key Format

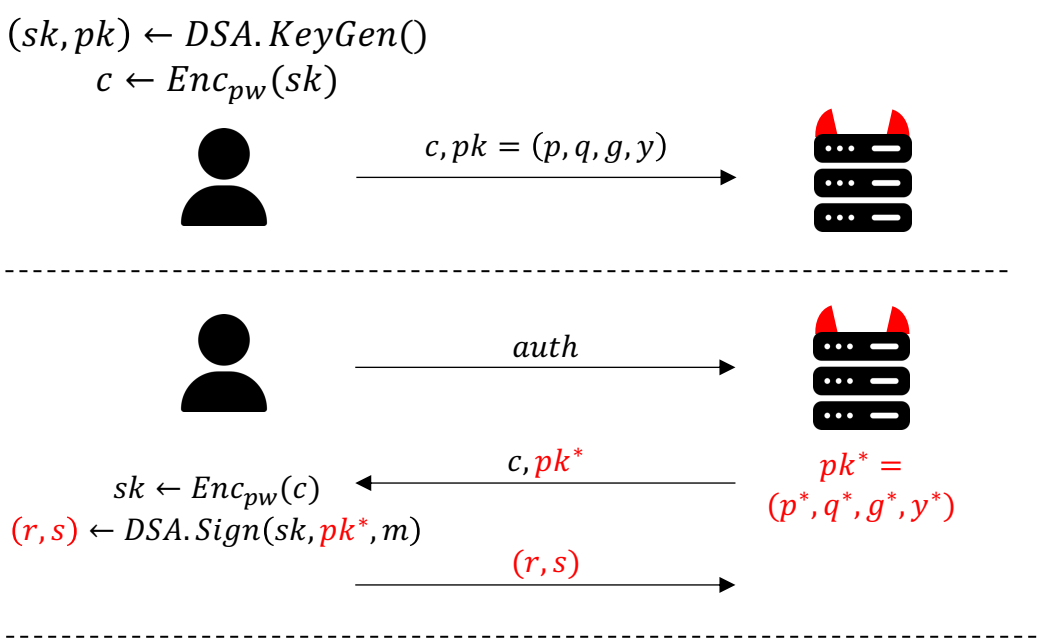
- No cryptographic protection for white fields
- Confidentiality for gray fields
  - (hash-then-encrypt instead of proper AE, so limited integrity)



Klíma and Rosa Attack on OpenPGP [1]

- Server can tamper with public key pk → pk^*
- Observe signatures (r, s) with bogus pk^* and correct sk

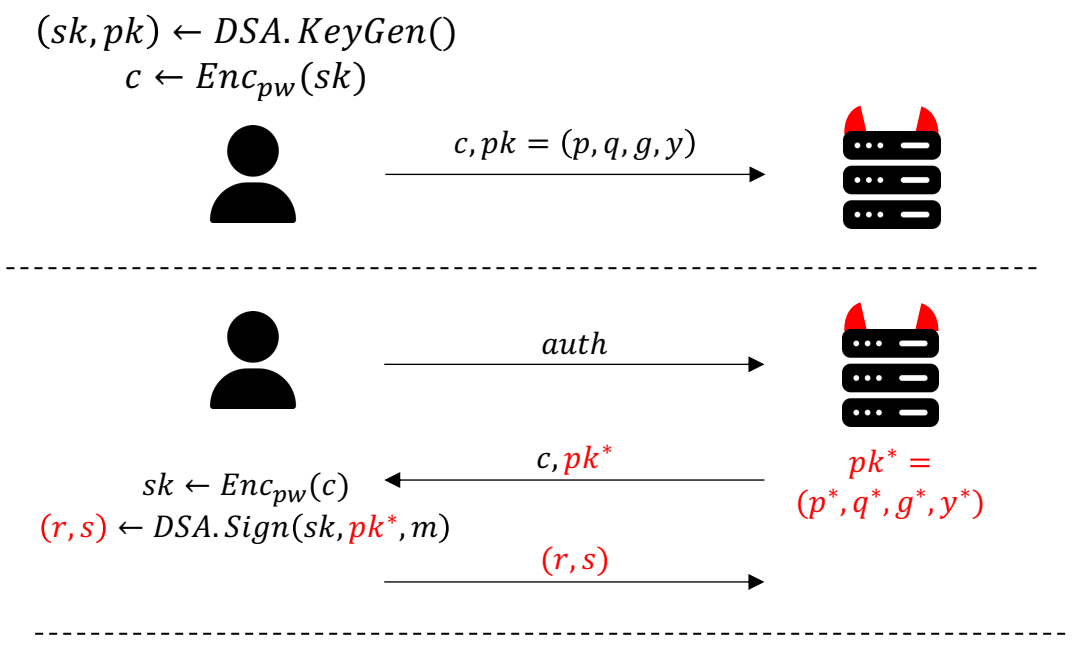
Q: What could possibly go wrong?
- Hint 1: think about previous attacks (L.9, L.12)
- Hint 2: security of DSA is based on hardness of DLOG



[1] Vlastimil Klíma and Tomas Rosa. “Attack on Private Signature Keys of the OpenPGP Format, PGP Programs and Other Applications Compatible with OpenPGP.” 2002.

Klíma and Rosa Attack on OpenPGP [1]

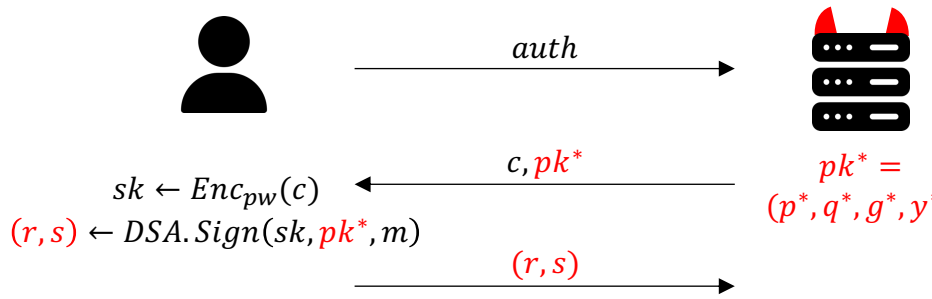
- Key overwriting
  - p^* prime such that $p^* - 1$ is smooth
  - $q^* > p^*$ (normally, $ord(g^*) \leq p^* - 1$)
  - g^* generator of $(\mathbb{Z}_{p^*})^*$
- Given (r, s), recover k:
  - $r = g^{*k} \bmod p^* \bmod q^* = g^{*k} \bmod p^*$
  - $k = \log_{g^*} g^{*k} \bmod p^*$
- Recover secret key sk
  - $sk = (s \cdot k - H(m))r^{-1} \bmod q$



[1] Vlastimil Klíma and Tomas Rosa. "Attack on Private Signature Keys of the OpenPGP Format, PGP Programs and Other Applications Compatible with OpenPGP." 2002.

Klíma and Rosa Attack [1]: Summary

- $n = 1$ attack
- Mitigation
  - Easily detectable: q^* is huge (typically only 160–256 bits)
  - Validating (sk, pk) would detect it (but optional in OpenPGP)
- Most libraries are not vulnerable
  - (parameter size restrictions)



$pk^* = (p^*, q^*, g^*, y^*)$ such that
- $p^* - 1$ is smooth
- $q^* > p^*$
- g^* generator of $(\mathbb{Z}_{p^*})^*$

key overwriting

Given (r, s):
- $k = \log_{g^*} g^{*k} \ mod \ p^*$
- $sk = \big(s \cdot k \ - H(m)\big) r^{-1} \ mod \ q$

key recovery

[1] Vlastimil Klíma and Tomas Rosa. “Attack on Private Signature Keys of the OpenPGP Format, PGP Programs and Other Applications Compatible with OpenPGP.” 2002.

Victory by KO [2]

- Bringing KO attacks on OpenPGP from 2002 to 2022

Victory by KO: Attacking OpenPGP Using Key Overwriting*

Lara Bruseghini
ETH Zurich and Proton AG
larabr@protonmail.com

Kenneth G. Paterson
Applied Cryptography Group, ETH Zurich
kenny.paterson@inf.ethz.ch

Daniel Huigens
Proton AG
d.huigens@protonmail.com

ABSTRACT

We present a set of attacks on the OpenPGP specification and implementations of it which result in full recovery of users' private keys. The attacks exploit the lack of cryptographic binding between the different fields inside an encrypted private key packet, which include the key algorithm identifier, the cleartext public parameters, and the encrypted private parameters. This allows an attacker who can overwrite certain fields in OpenPGP key packets to perform

downloading and sending messages, and remote parties do not get to communicate directly with cryptographic software, but where that software is only used locally to decrypt/encrypt or sign/verify some emails. However, the use cases for OpenPGP have evolved, and application scenarios have changed over the past 20 years. In particular, we now see widespread use of cloud-based storage, in-browser and server-provided encryption services, and automated cryptographic processing by those services. Hence, modelling as-

- Do Klíma and Rosa attack *without* huge q?

[2] Lara Bruseghini, Daniel Huigens, Kenneth G. Paterson."Victory by KO: Attacking OpenPGP Using Key Overwriting". CCS 2022. (link)

Victory by KO [2]

- Key overwriting
  - p^* prime such that $q_i \mid (p^* - 1)$
  - $q_i \ll p^*$ (~16 bits)
  - g^* generator with order q_i
- Recover $\mathrm{x_i} = \mathrm{x}\ mod\ q_i$ (whp):
  - for $\mathrm{x_i} = 0, 1, \ldots, q_i - 1$:
    - $pk = (p^*, q_i, g^*, y = (g^*)^{x_i}\ mod\ p)$
    - if $Verify(pk, m, (r, s))$: return $\mathrm{x_i}$

Q: How do we recover x?

$Verify(pk, m, (r, s))$:
$u_1 = H(m)s^{-1}\ mod\ q$
$u_2 = rs^{-1}\ mod\ q$
return $r = g^{u_1}y^{u_2}\ mod\ p\ mod\ q$

$Sign(sk = x, pk = (\mathrm{p}, \mathrm{q}, \mathrm{g}, \mathrm{y}), m)$:
$k \leftarrow_{\$} \mathbb{Z}_q$
$r = g^k\ mod\ p\ mod\ q$
$s = k^{-1}(H(m) + xr)\ mod\ q$
return (r, s)

[2] Lara Bruseghini, Daniel Huigens, Kenneth G. Paterson."Victory by KO: Attacking OpenPGP Using Key Overwriting". CCS 2022. (link)

Victory by KO [2]

- Key overwriting
  - p^* prime such that $q_i \mid (p^* - 1)$
  - $q_i \ll p^*$ (~16 bits)
  - g^* generator with order q_i
- Recover $\mathrm{x_i} = \mathrm{x}\ mod\ q_i$ (whp):
  - for $\mathrm{x_i} = 0, 1, \ldots, q_i - 1$:
    - $pk = (p^*, q_i, g^*, y = (g^*)^{x_i}\ mod\ p)$
    - `if` $Verify(pk, m, (r, s))$: `return` $\mathrm{x_i}$
- Recover x with CRT from enough samples $\mathrm{x_i} = \mathrm{x}\ mod\ q_i$ for $i = 1, \ldots, n$

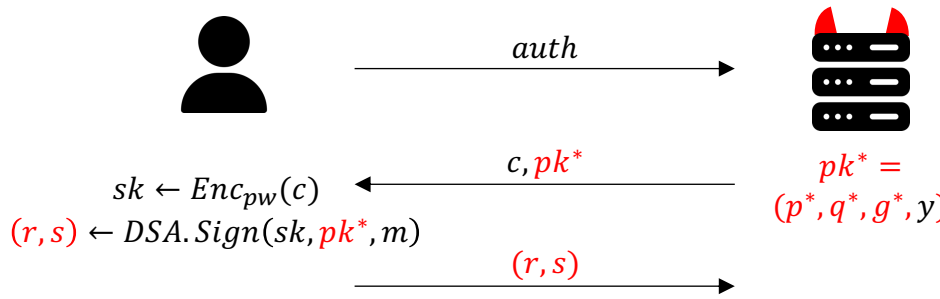
$Verify(pk, m, (r, s))$:
$u_1 = H(m)s^{-1}\ mod\ q$
$u_2 = rs^{-1}\ mod\ q$
`return` $r = g^{u_1}y^{u_2}\ mod\ p\ mod\ q$

$Sign(sk = x, pk = (\mathrm{p}, \mathrm{q}, \mathrm{g}, \mathrm{y}), m)$:
$k \leftarrow_{\$} \mathbb{Z}_q$
$r = g^k\ mod\ p\ mod\ q$
$s = k^{-1}(H(m) + xr)\ mod\ q$
return (r, s)

[2] Lara Bruseghini, Daniel Huigens, Kenneth G. Paterson."Victory by KO: Attacking OpenPGP Using Key Overwriting". CCS 2022. (link)

Victory by KO [2]: summary

- Recovers secret key with n interactions
- Much more in paper:
  - Cross-algorithm attack
  - Attack exploiting key validation (KOKV-KR)
  - More attacks
  - Mitigations



$pk^* = (p^*, q^*, g^*, y^*)$ such that
- p^* prime such that $q_i \mid (p^* - 1)$
- $q_i \ll p^*$
- g^* generator with order q_i

key overwriting

- Recover $x_i = x \bmod q_i$:
  - Try $Verify(pk, m, (r,s))$ for all x_i
  - W.h.p. succeeds only for correct x_i
- Recover x with CRT

key recovery

[2] Lara Bruseghini, Daniel Huigens, Kenneth G. Paterson."Victory by KO: Attacking OpenPGP Using Key Overwriting". CCS 2022. (link)

Attacks on OpenPGP: Questions?

- Questions?
  - On how great OpenPGP is designed?
  - On the Klíma and Rosa attack?
  - On the Victory by KO attacks?

- Up next: attacks on MEGA


Attacks on MEGA

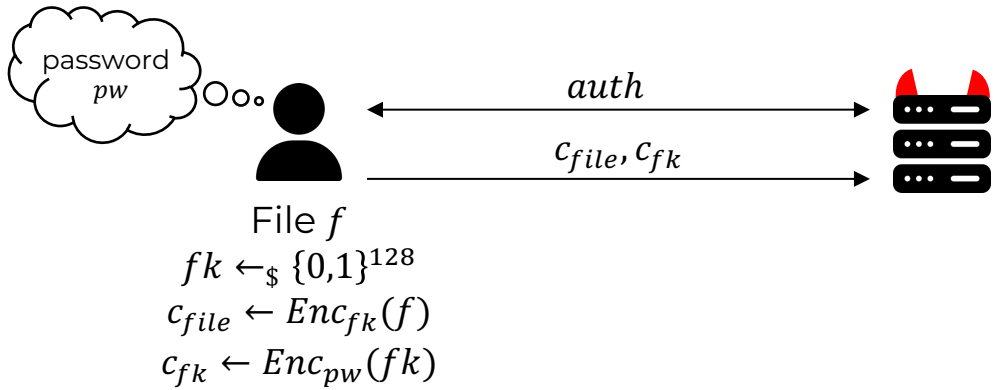
MEGA: Setting

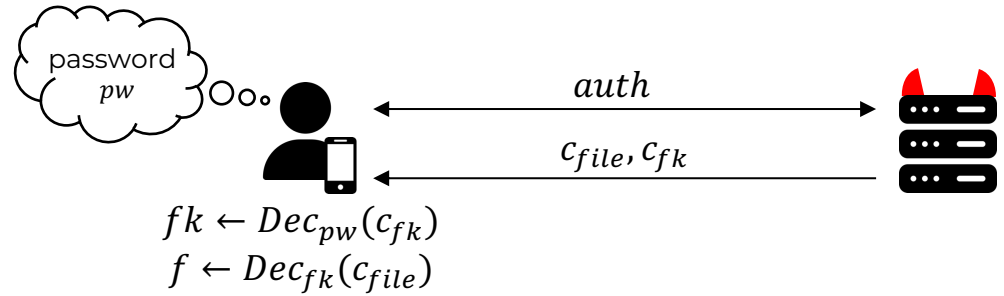
- The largest end-to-end encrypted cloud storage
  - Almost 300M accounts storing 140B+ files [3]
  - Claiming strong privacy, even against themselves

[3] https://mega.io/about (08/2023)

MEGA: Design–Files

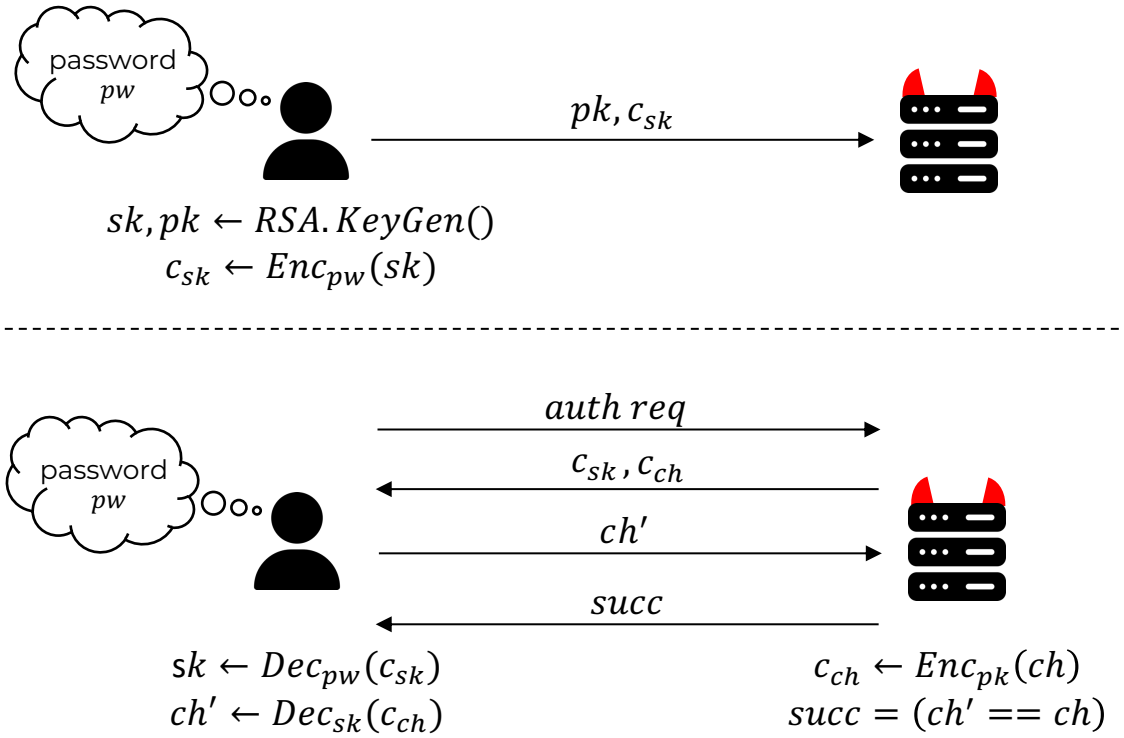
- Uploading file f
  - Authenticate
  - Pick fresh file key fk
  - Encrypt file f with fk
  - Encrypt key fk with pw
  - Upload both ciphertexts to server
- Downloading file f
  - Authenticate
  - Obtain ciphertexts c_{fk}, c_{file}
  - Decrypt file key fk with password
  - Decrypt file f with fk





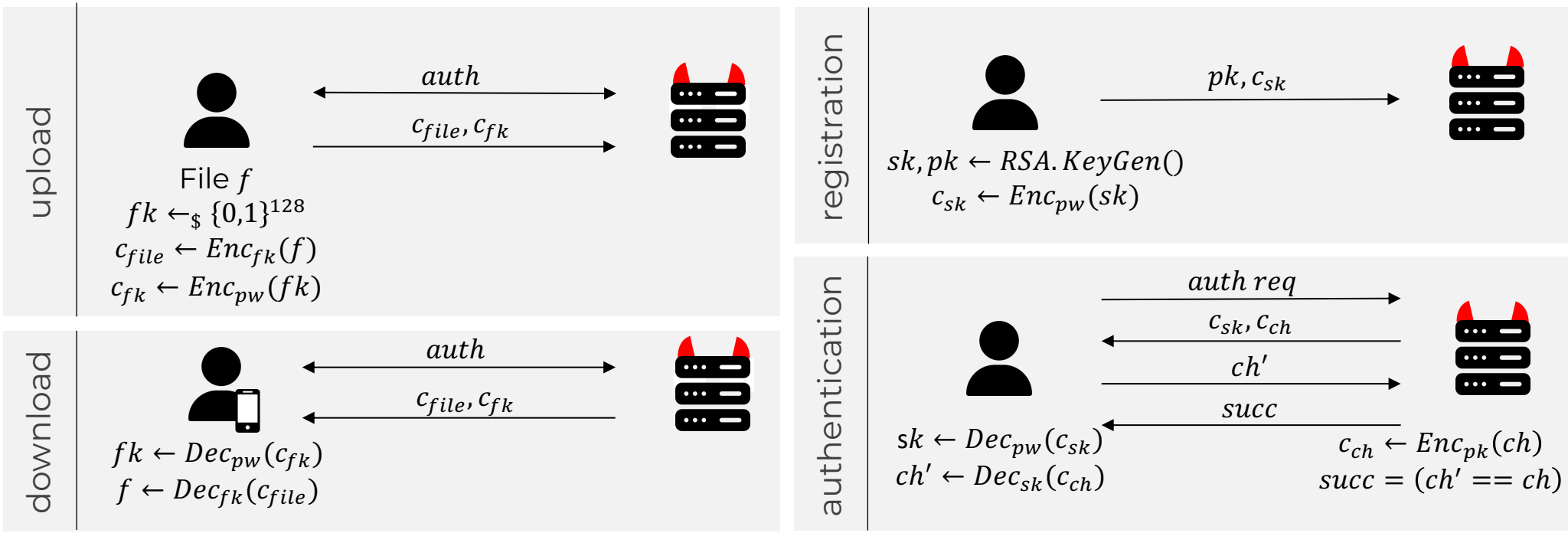
MEGA: Design–User Authentication

- Registration
  - Generate RSA key pair (sk, pk)
  - Encrypt secret key sk with pw
  - Upload pk and ciphertext c_{sk}
- Authentication $auth$
  - Client requests to authenticate
  - Server sends secret key ctxt c_{sk}
  - Server encrypts challenge for pk
  - Client recovers RSA sk using pw and decrypts challenge to ch'
  - Sends ch' back to server
  - Successful if ch equals ch'
    - shows knowledge of sk



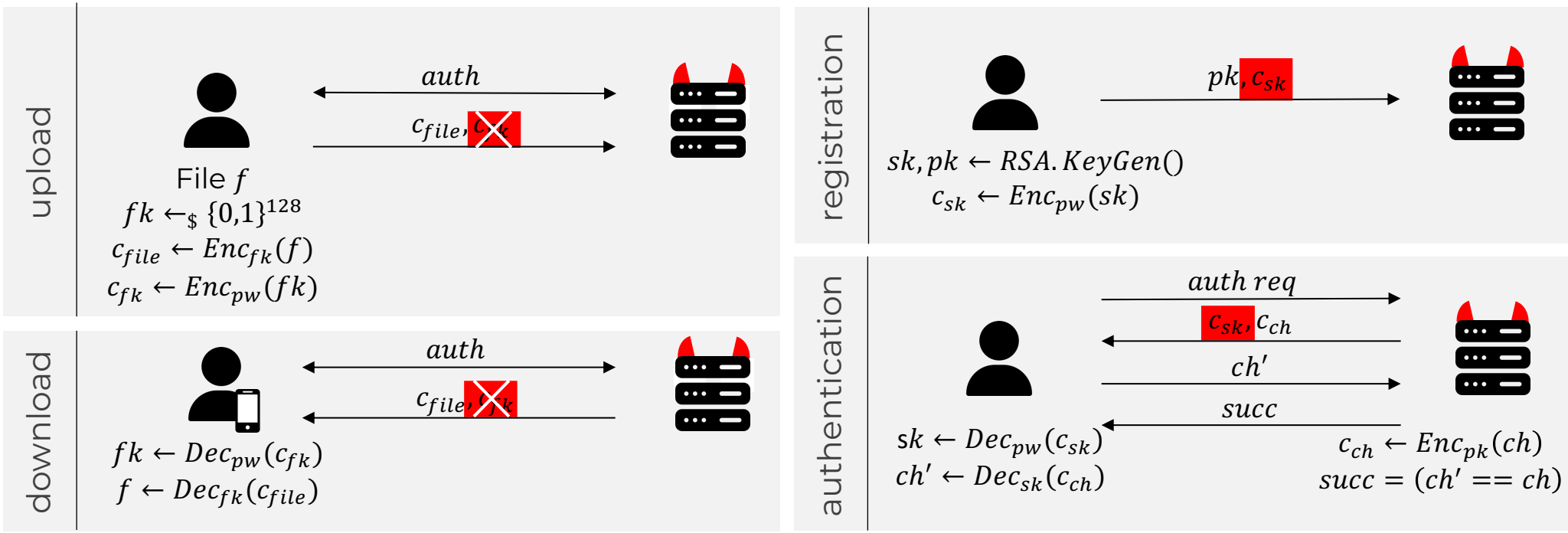
MEGA: Design flaws

Q: Where would you attack this system?
Hint: think about Key Overwriting attacks!

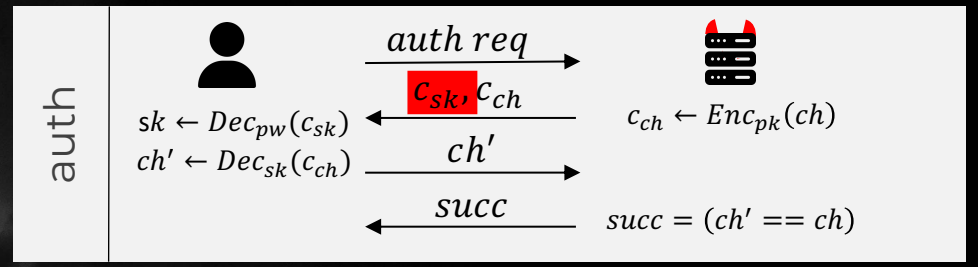


MEGA: Design flaws

A: Outsourced key ciphertexts!
File keys turn out to be too short, but sk...

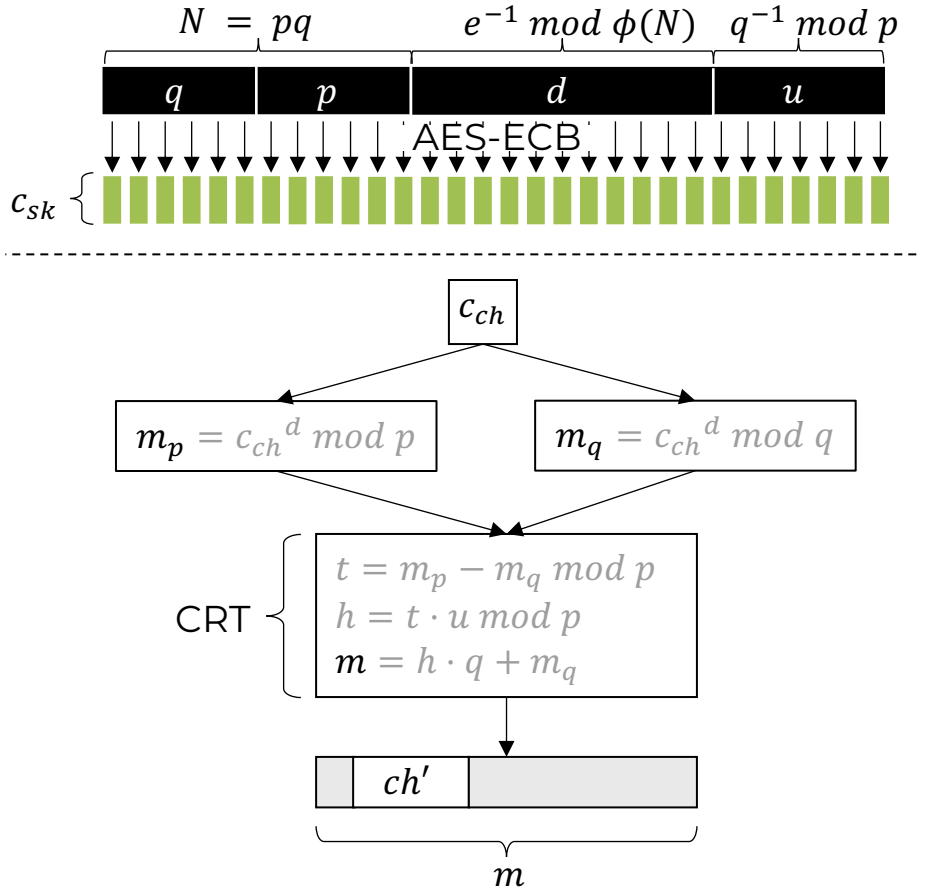


MEGA: Attack 1 [4]



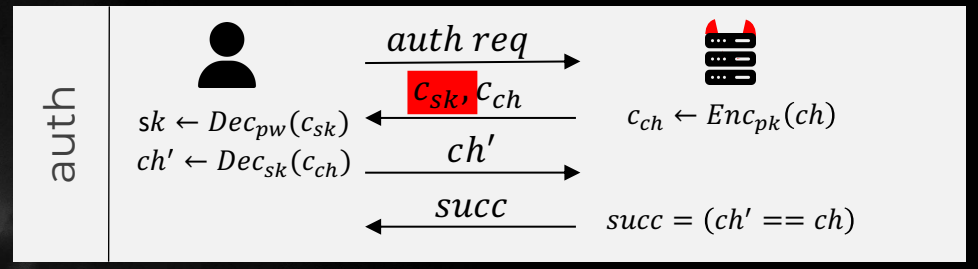
- RSA sk format
  - Primes p, q; secret exponent d;
  - u for RSA-CRT decryption
- RSA-CRT challenge decryption
  - Decrypt in $\mathbb{Z}_p$ and $\mathbb{Z}_q$
  - Reconstruct $m \in \mathbb{Z}_N$ with CRT
  - Remove padding to recover ch'

Q: How can we overwrite sk?
A: AES-ECB is malleable!

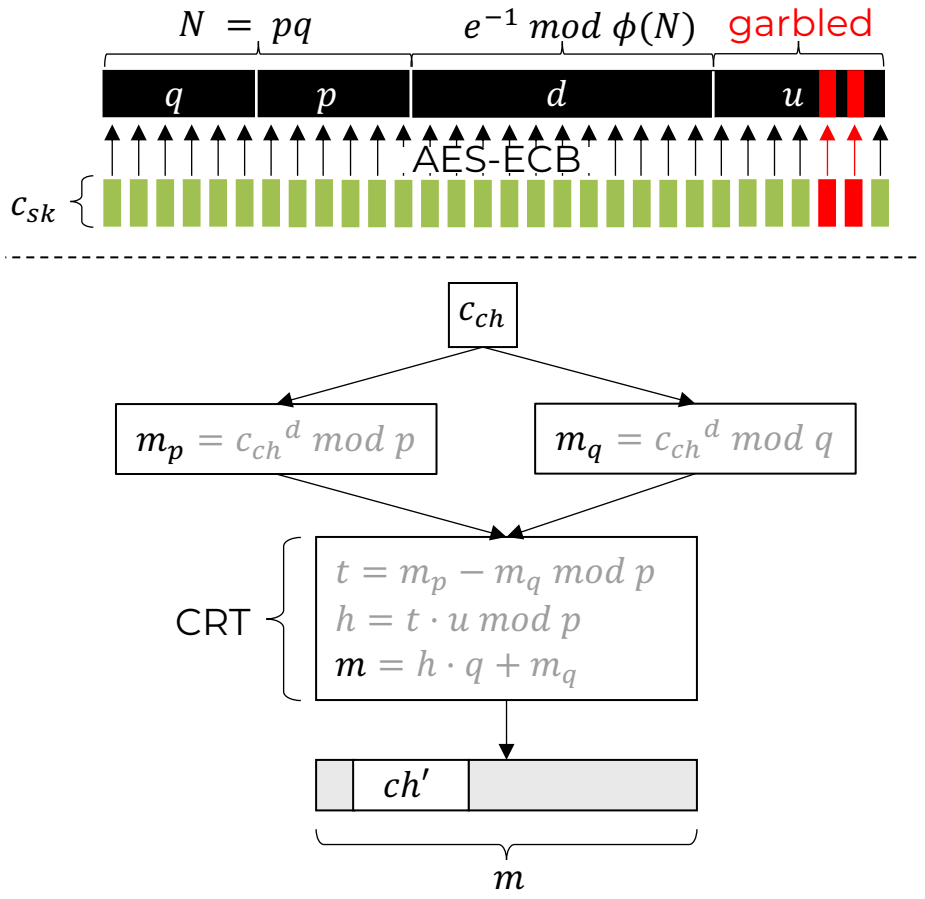


[4] Matilda Backendal, Miro Haller and Kenneth G. Paterson. “MEGA: Malleable Encryption Goes Awry”. IEEE S&P 2023.

MEGA: Attack 1 [4]

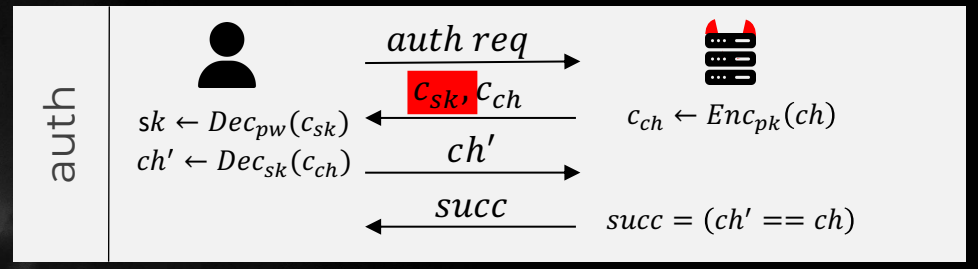


- Key overwriting
  - Overwriting some AES-ECB blocks garbles plaintext u
  - Without affecting q, p, d!

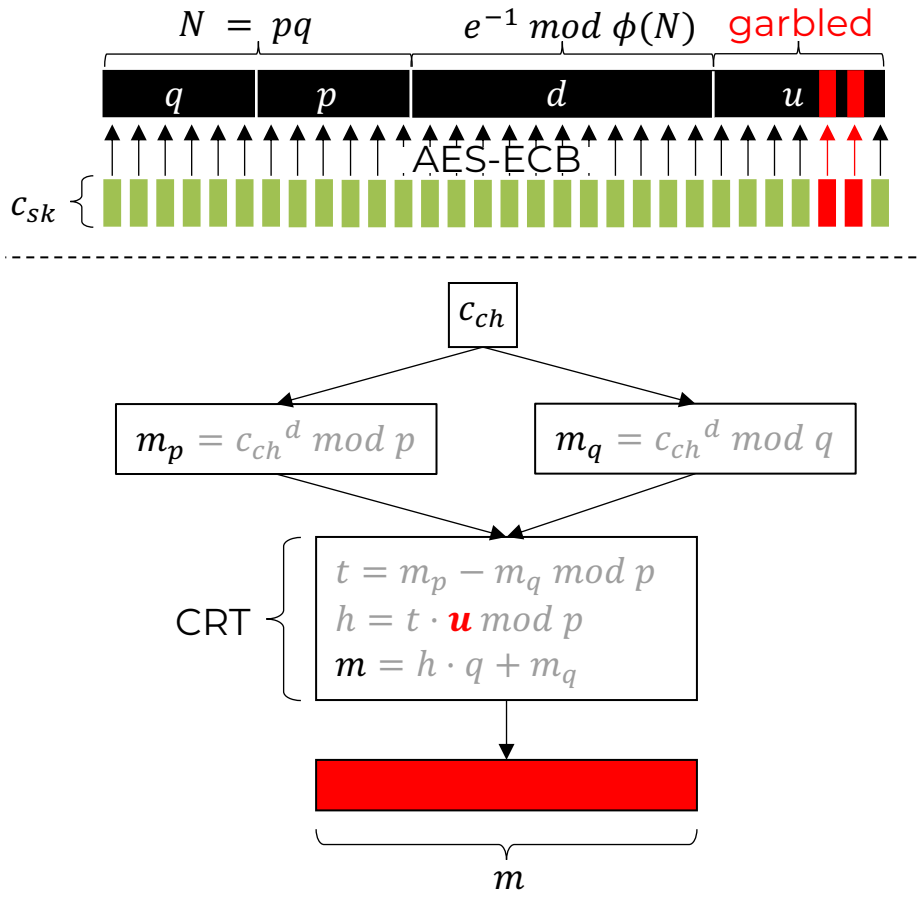


[4] Matilda Backendal, Miro Haller and Kenneth G. Paterson. “MEGA: Malleable Encryption Goes Awry”. IEEE S&P 2023.

MEGA: Attack 1 [4]

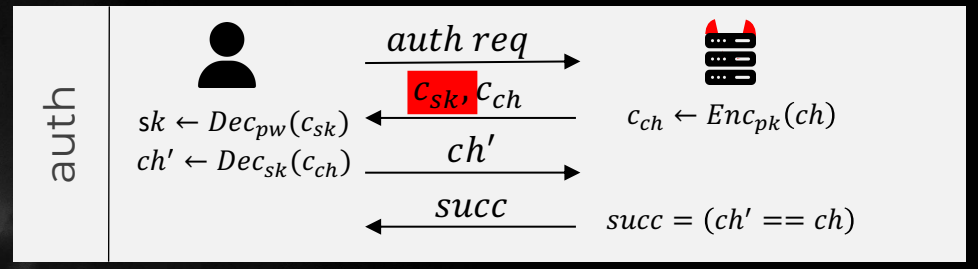


- Key overwriting
  - Overwriting some AES-ECB blocks garbles plaintext u
  - Without affecting q, p, d!
- In general, garbles decryption

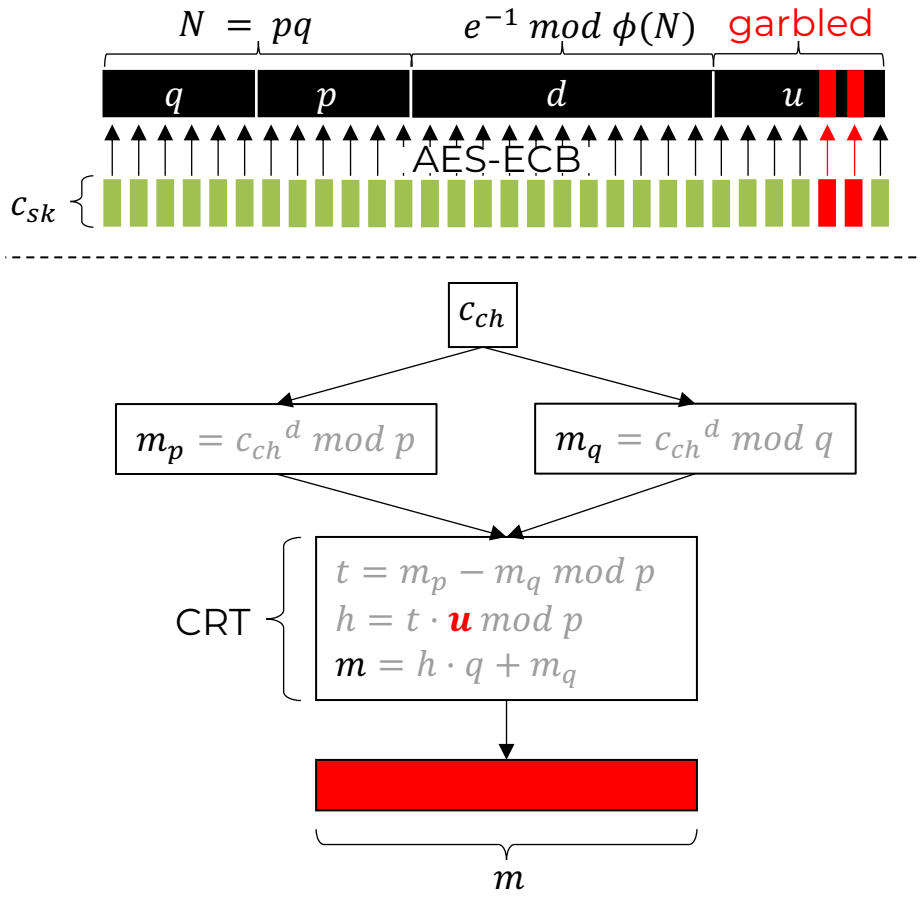


[4] Matilda Backendal, Miro Haller and Kenneth G. Paterson. “MEGA: Malleable Encryption Goes Awry”. IEEE S&P 2023.

MEGA: Attack 1 [4]

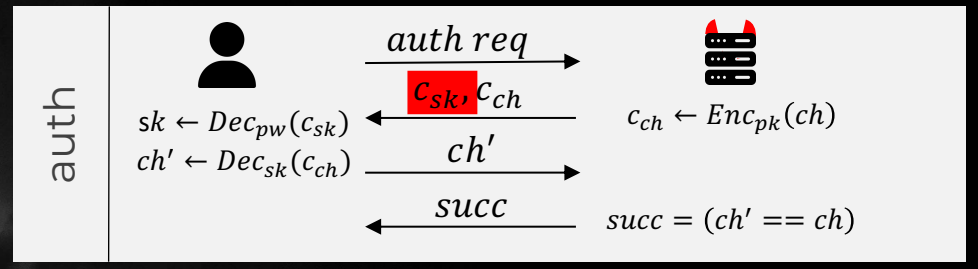


- Key overwriting
  - Overwriting some AES-ECB blocks garbles plaintext u
  - Without affecting q, p, d!
- In general, garbles decryption
- But still succeeds if $ch < p, q$

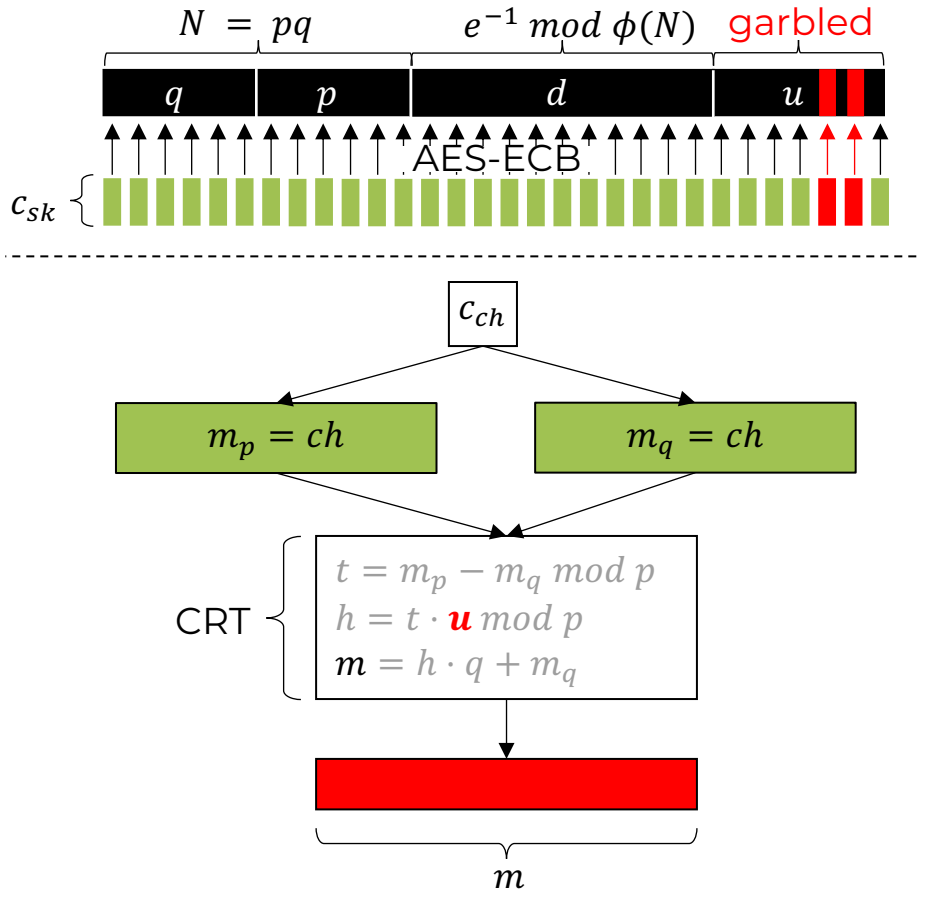


[4] Matilda Backendal, Miro Haller and Kenneth G. Paterson. “MEGA: Malleable Encryption Goes Awry”. IEEE S&P 2023.

MEGA: Attack 1 [4]

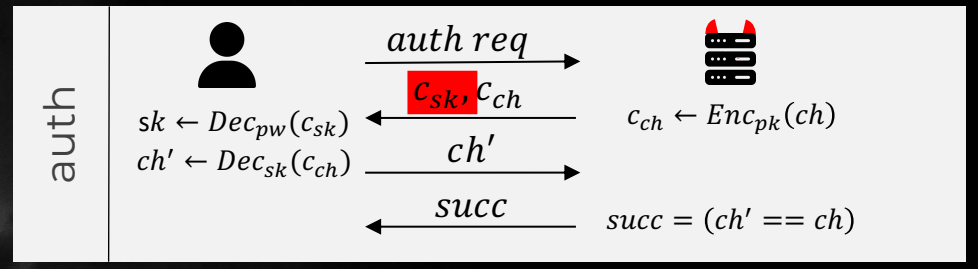


- Key overwriting
  - Overwriting some AES-ECB blocks garbles plaintext u
  - Without affecting q, p, d!
- In general, garbles decryption
- But still succeeds if $ch < p, q$
  - $m_p = m_q = ch$ because decryption in $\mathbb{Z}_p$ and $\mathbb{Z}_q$ is unique!

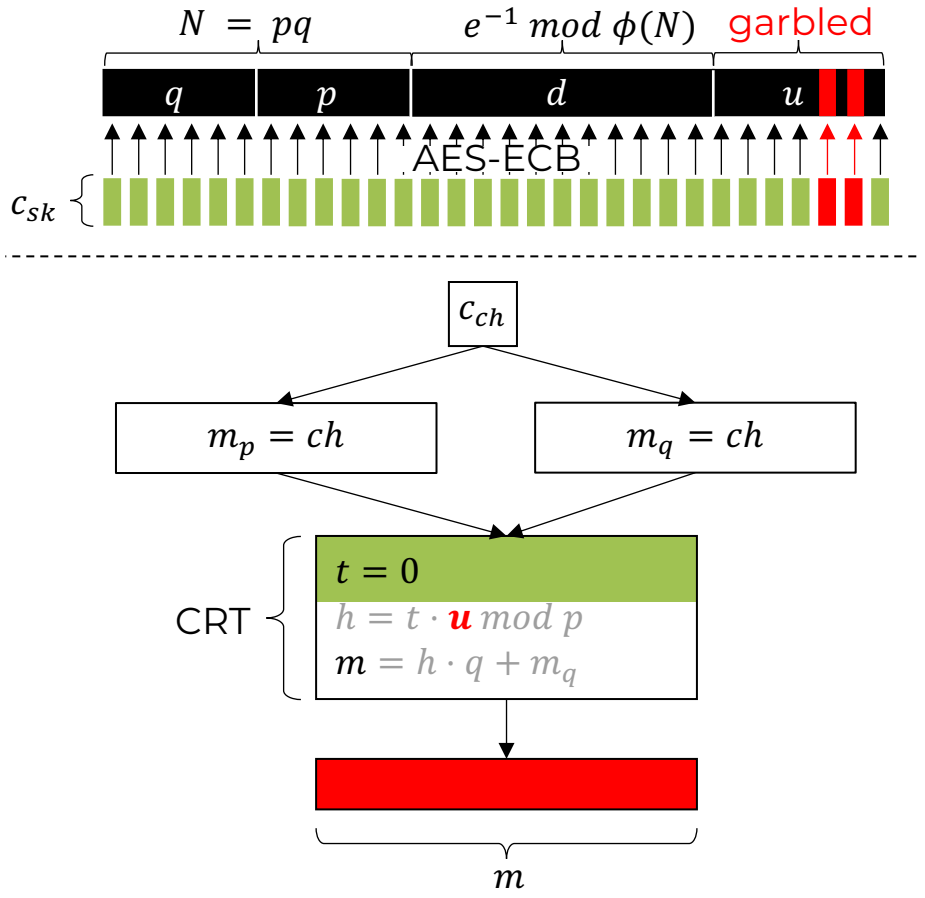


[4] Matilda Backendal, Miro Haller and Kenneth G. Paterson. “MEGA: Malleable Encryption Goes Awry”. IEEE S&P 2023.

MEGA: Attack 1 [4]

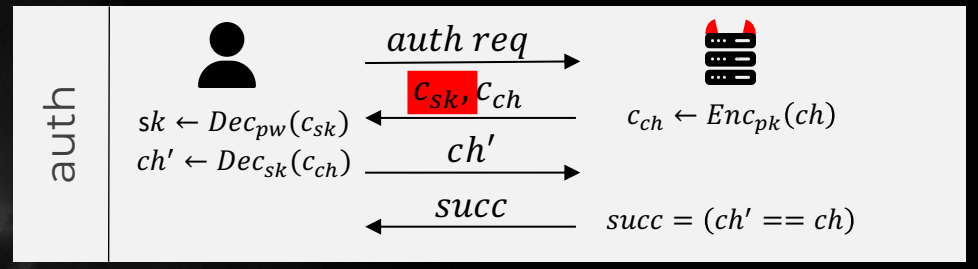


- Key overwriting
  - Overwriting some AES-ECB blocks garbles plaintext u
  - Without affecting q, p, d!
- In general, garbles decryption
- But still succeeds if $ch < p, q$
  - $m_p = m_q = ch$ because decryption in $\mathbb{Z}_p$ and $\mathbb{Z}_q$ is unique!
  - Simplifies CRT

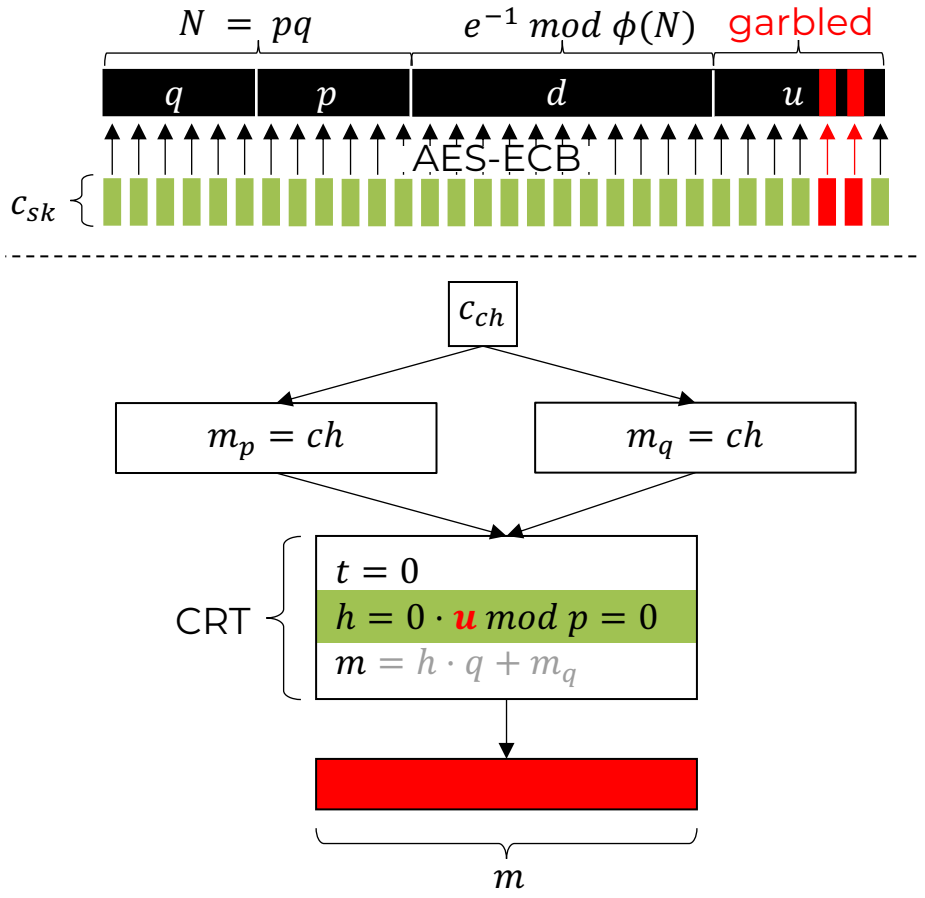


[4] Matilda Backendal, Miro Haller and Kenneth G. Paterson. “MEGA: Malleable Encryption Goes Awry”. IEEE S&P 2023.

MEGA: Attack 1 [4]

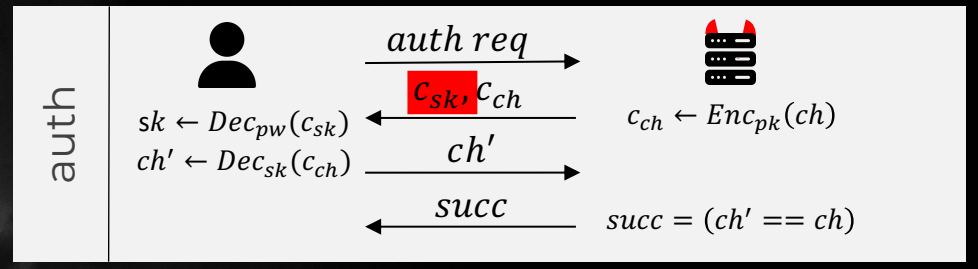


- Key overwriting
  - Overwriting some AES-ECB blocks garbles plaintext u
  - Without affecting q, p, d!
- In general, garbles decryption
- But still succeeds if $ch < p, q$
  - $m_p = m_q = ch$ because decryption in $\mathbb{Z}_p$ and $\mathbb{Z}_q$ is unique!
  - Simplifies CRT

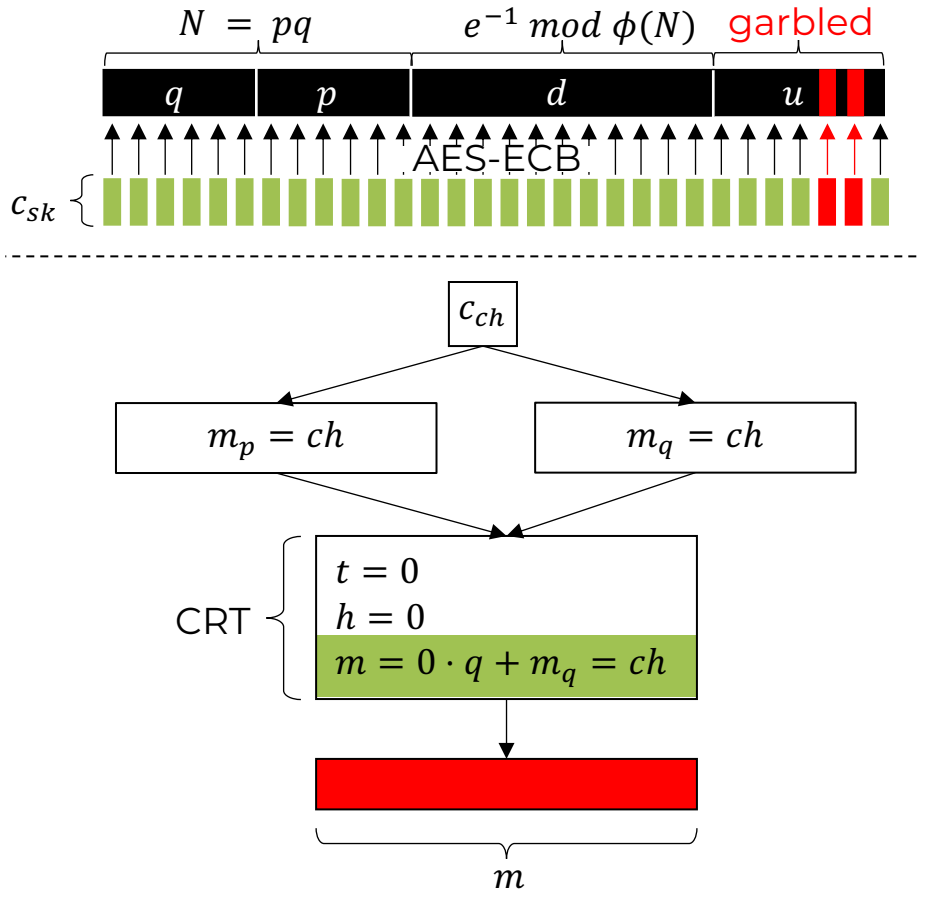


[4] Matilda Backendal, Miro Haller and Kenneth G. Paterson. “MEGA: Malleable Encryption Goes Awry”. IEEE S&P 2023.

MEGA: Attack 1 [4]

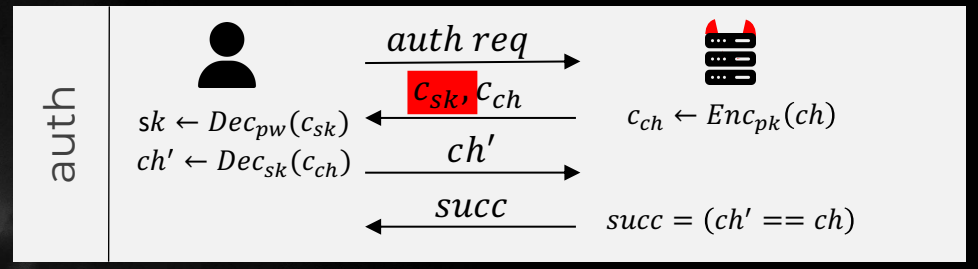


- Key overwriting
  - Overwriting some AES-ECB blocks garbles plaintext u
  - Without affecting q, p, d!
- In general, garbles decryption
- But still succeeds if $ch < p, q$
  - $m_p = m_q = ch$ because decryption in $\mathbb{Z}_p$ and $\mathbb{Z}_q$ is unique!
  - Simplifies CRT

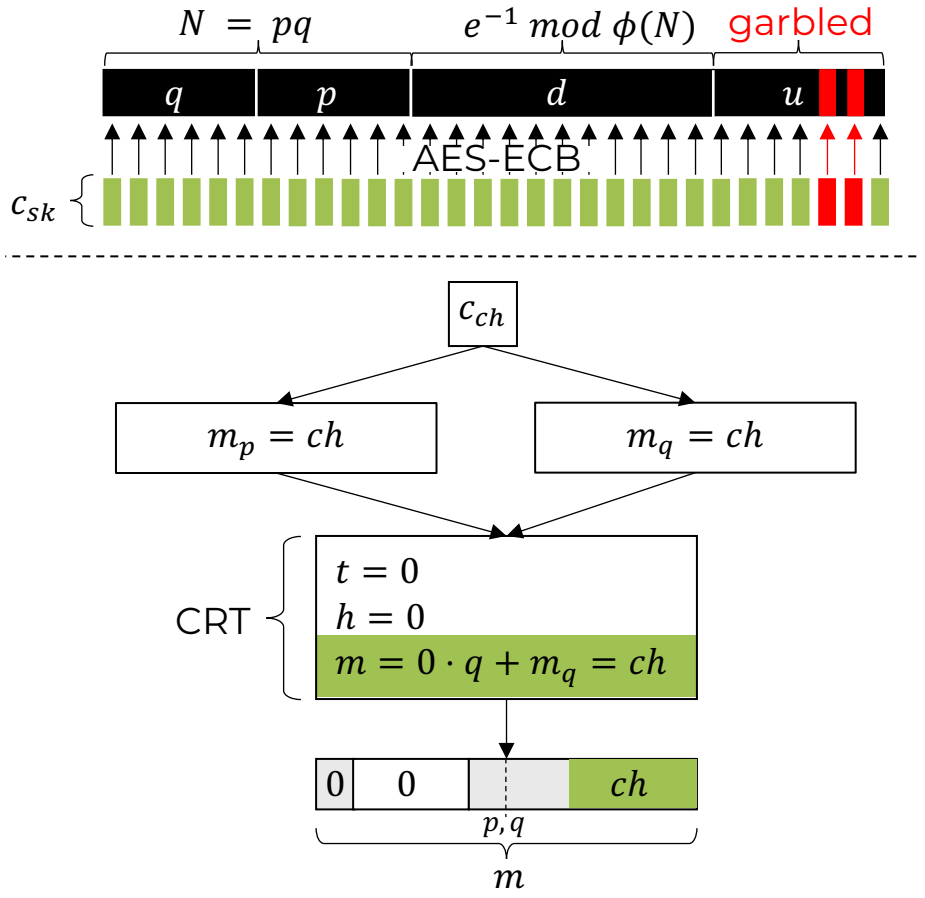


[4] Matilda Backendal, Miro Haller and Kenneth G. Paterson. “MEGA: Malleable Encryption Goes Awry”. IEEE S&P 2023.

MEGA: Attack 1 [4]

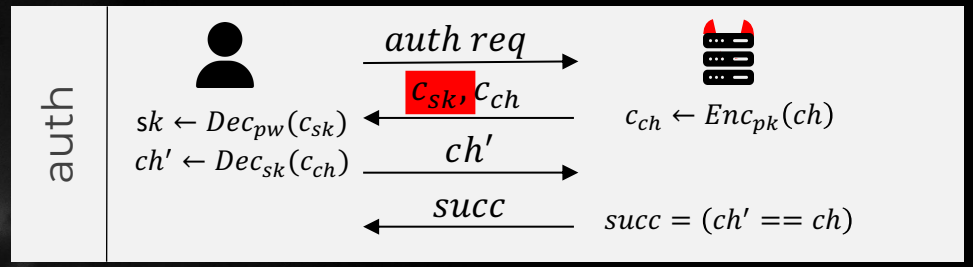


- Key overwriting
  - Overwriting some AES-ECB blocks garbles plaintext u
  - Without affecting q, p, d!
- In general, garbles decryption
- But still succeeds if $ch < p, q$
  - $m_p = m_q = ch$ because decryption in $\mathbb{Z}_p$ and $\mathbb{Z}_q$ is unique!
  - Simplifies CRT



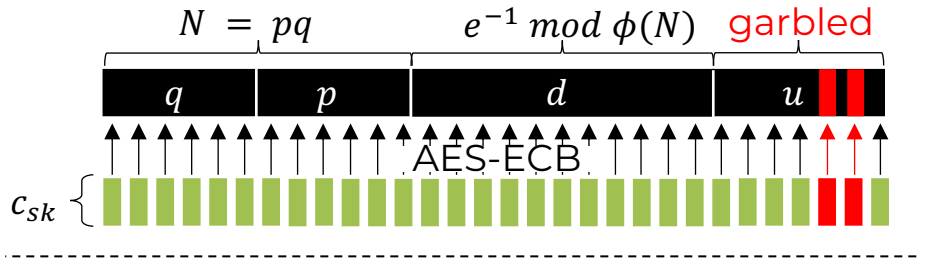
[4] Matilda Backendal, Miro Haller and Kenneth G. Paterson. “MEGA: Malleable Encryption Goes Awry”. IEEE S&P 2023.

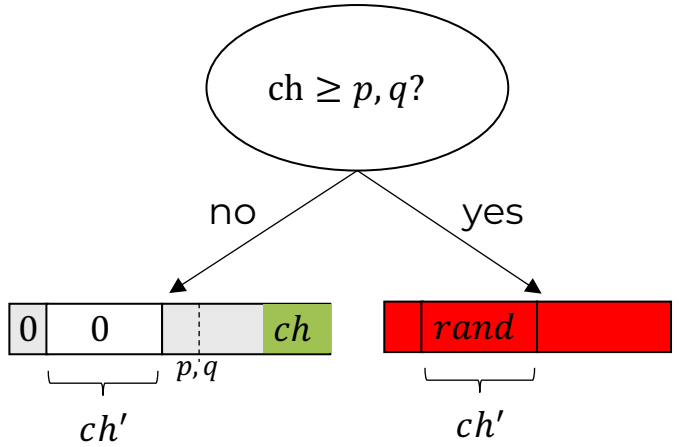
MEGA: Attack 1 [4]



- In summary
  - Overwrite AES-ECB blocks
  - If $ch \geq p, q$ then $ch' = rand$
  - If $ch < p, q$ then $ch' = 0$

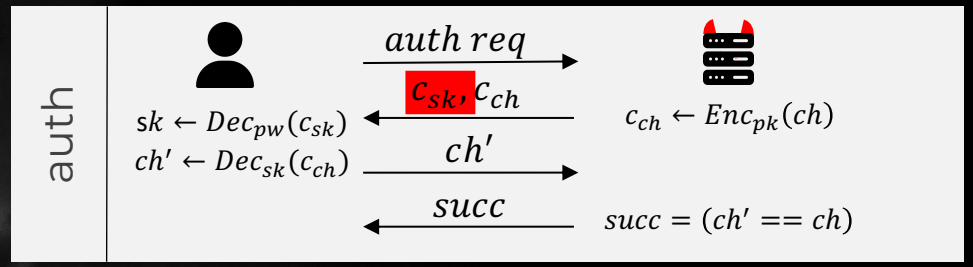
Q: How do we recover sk?



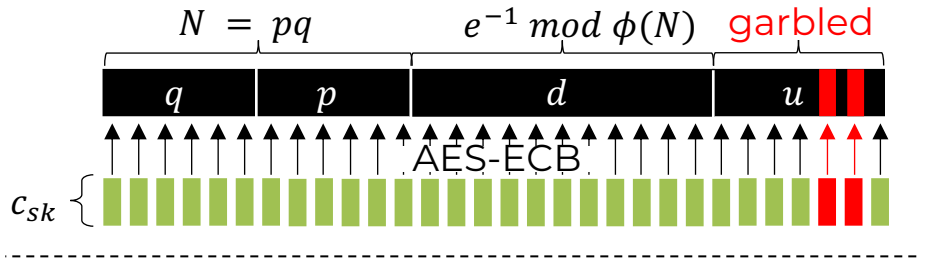


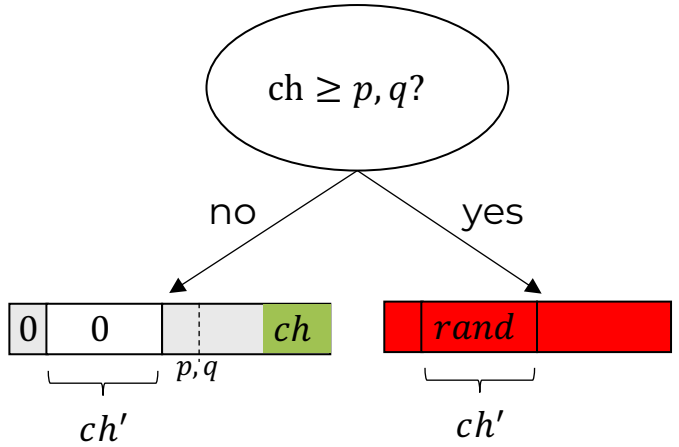
[4] Matilda Backendal, Miro Haller and Kenneth G. Paterson. “MEGA: Malleable Encryption Goes Awry”. IEEE S&P 2023.

MEGA: Attack 1 [4]



- In summary
  - Overwrite AES-ECB blocks
  - If $ch \geq p, q$ then $ch' = rand$
  - If $ch < p, q$ then $ch' = 0$
- Recover sk with binary search
  - Requires 512 login attempts
  - Later improved to 6 [5] and 2 [6]
- Details and 4 more attacks in [4]
  - Together, achieve file decryption

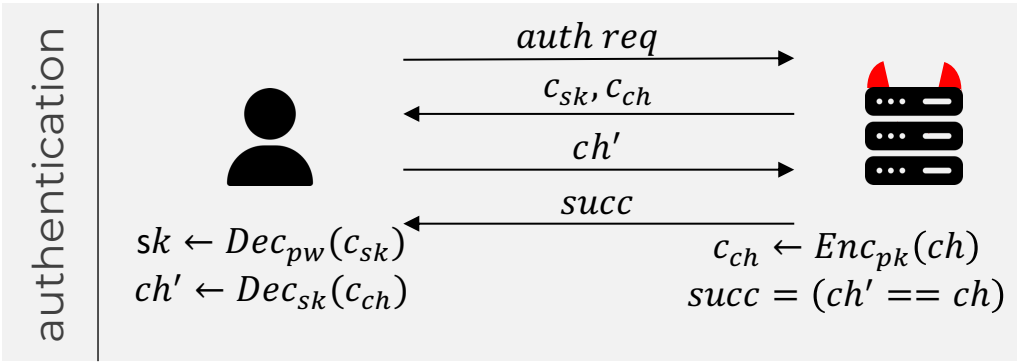




[4] Matilda Backendal, Miro Haller and Kenneth G. Paterson. “MEGA: Malleable Encryption Goes Awry”. IEEE S&P 2023.
[5] Ryan, Keegan, and Heninger, Nadia. "The Hidden Number Problem with Small Unknown Multipliers: Cryptanalyzing MEGA in Six Queries and Other Applications." Public-Key Cryptography. 2023.
[6] Martin R. Albrecht, Miro Haller, Lenka Mareková, Kenneth G. Paterson. “Caveat Implementor! Key Recovery Attacks on MEGA”. Eurocrypt 2023.

MEGA: Breaking the Patch [6]

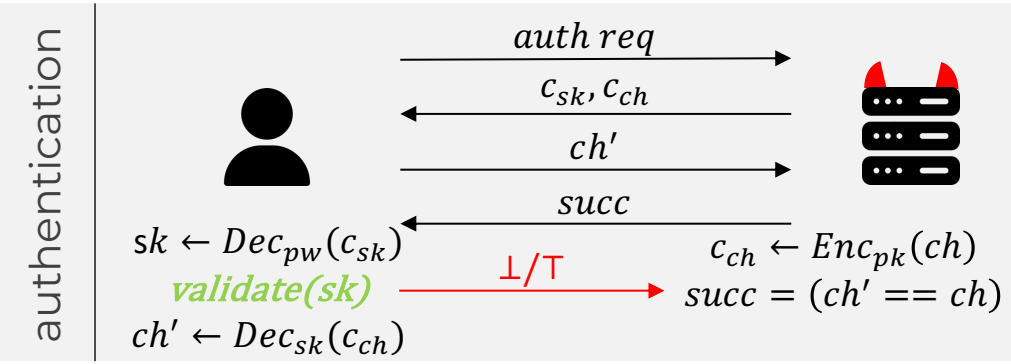
Q: If you were MEGA, how would you mitigate this attack?



[6] Martin R. Albrecht, Miro Haller, Lenka Mareková, Kenneth G. Paterson. “Caveat Implementor! Key Recovery Attacks on MEGA”. Eurocrypt 2023.

MEGA: Breaking the Patch [6]

- Mitigation:
  - Validate secret key format
    - E.g., verifying $\mathrm{u} == q^{-1}\ mod\ p$
  - Not protecting integrity of c_{sk}
- Clients leak if sk is valid
  - Error oracle attack in [6]

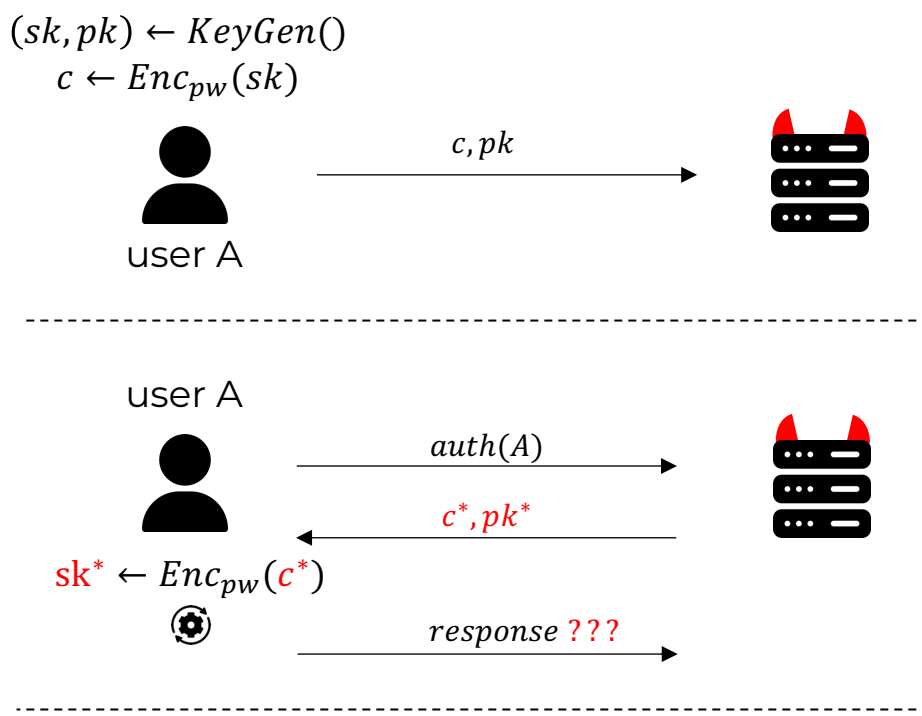


[6] Martin R. Albrecht, Miro Haller, Lenka Mareková, Kenneth G. Paterson. “Caveat Implementor! Key Recovery Attacks on MEGA”. Eurocrypt 2023.

Summary:
Key Overwriting Attacks

Summary

- Threat model:
  - Malicious server stores encrypted key material of client
  - Runs interactive protocol with client
- Malicious server…
  - … may override key ciphertexts
  - … aims to recover the key
- Examples in the wild:
  - OpenPGP: Klíma and Rosa, Victory by KO
  - MEGA: RSA key recovery attack, error oracle, and more



UCSD Evaluations

- Evaluate TA performance
- Anonymous
  - we get aggregated results after final grades
- But feedback should be constructive (UCSD principles)

src: https://tvtropes.org/pmwiki/pmwiki.php/main/HoldUpYourScore (11/29/2023)

https://academicaffairs.ucsd.edu/Modules/Evals?e11061127

Additional Resources

Additional Resources

- Icons from flaticon.com:
  - Freepik: user, server, systems update, cell phone