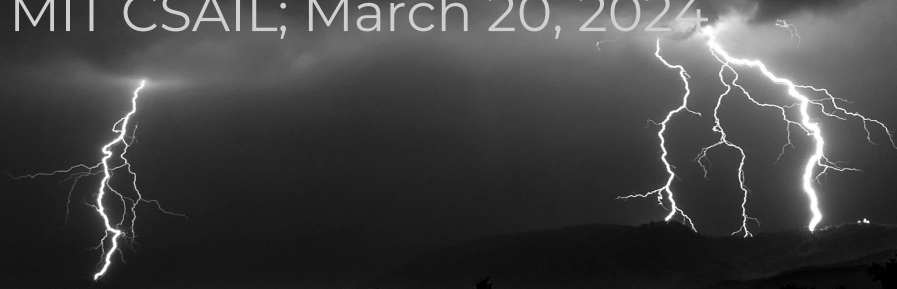





Key Overwriting Attacks

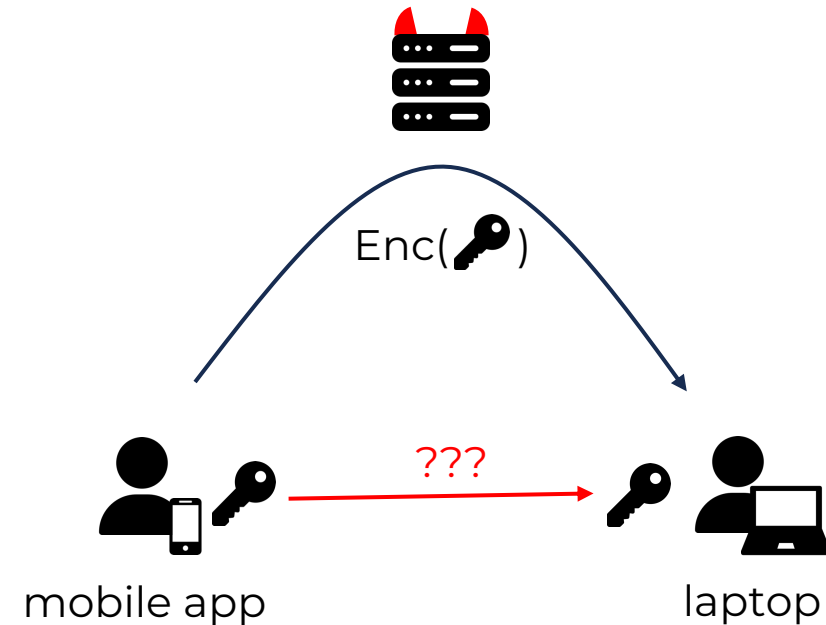
Miro Haller

MIT CSAIL; March 20, 2024



E2EE Shifts Threat Model

- More E2EE applications, e.g.:
 - Secure messaging 
 - Encrypted email  Proton Mail
 - Cloud storage  MEGA
- Client-side encryption
 - Keys on client
 - Transfer over server
- Attacks on Enc()!



KO Attacks: Informal Definition

Definition [Key Overwriting (KO) Attacks]:

who?

A malicious service provider attacks a user

how?

while executing an interactive protocol and
by *overwriting* encrypted key material that
the user previously stored on the server

why?

with the goal to break confidentiality or
recover the user's key material.

Attacks In The Wild

[\[eprint:KR02\]](#)

[\[CCS:BHP22\]](#)



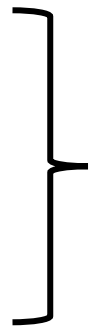
OpenPGP:  **Proton Mail**

- standard for email encryption
- E.g., protonmail w/ 100M+ users

[\[IEEESP:BHP23\]](#)

[\[Eurocrypt:AHMP23\]](#)

[\[PKC:RH23\]](#)



MEGA:  **MEGA**

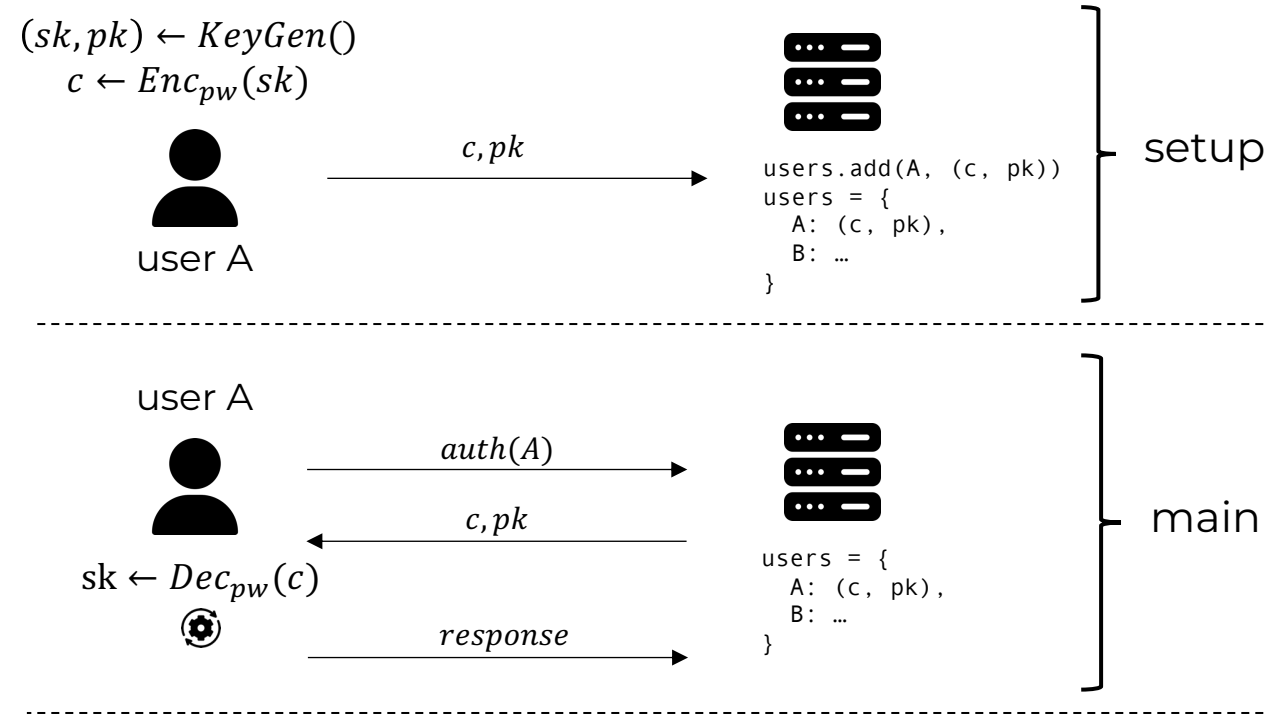
- E2EE cloud storage provider
- 300M+ users

Formalizing KO Attacks



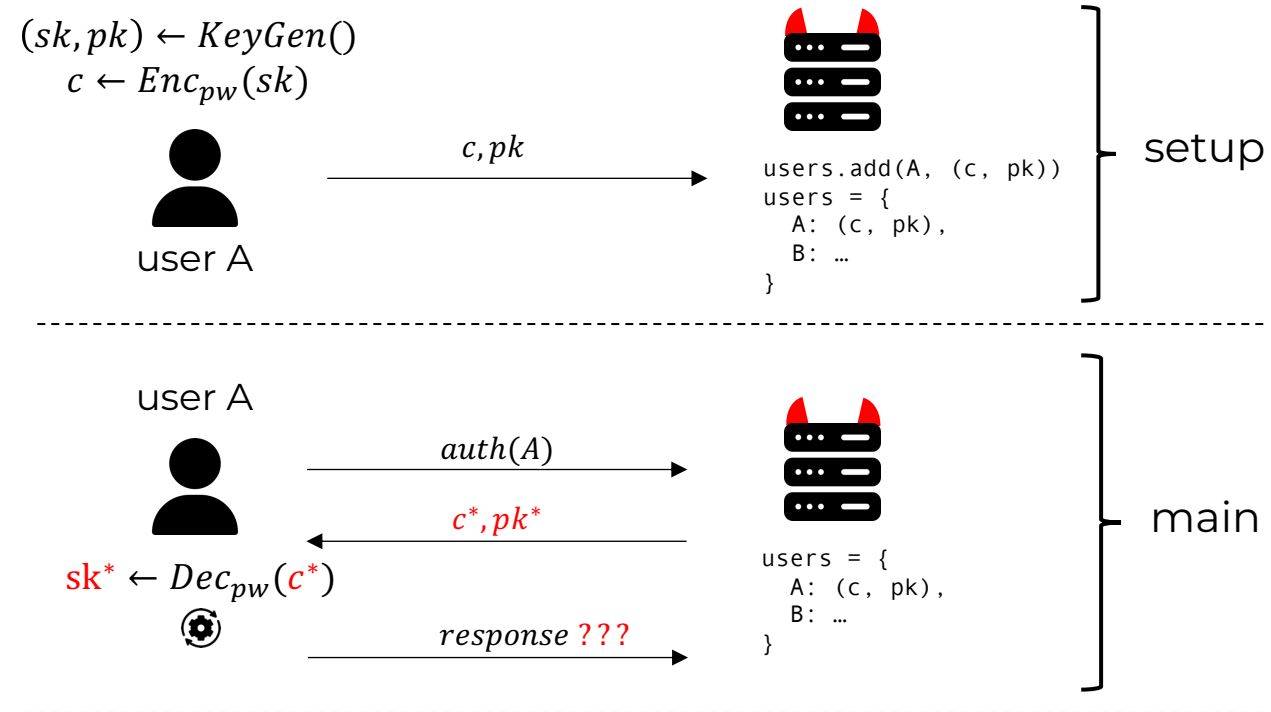
KO Attacks: Setting

- 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



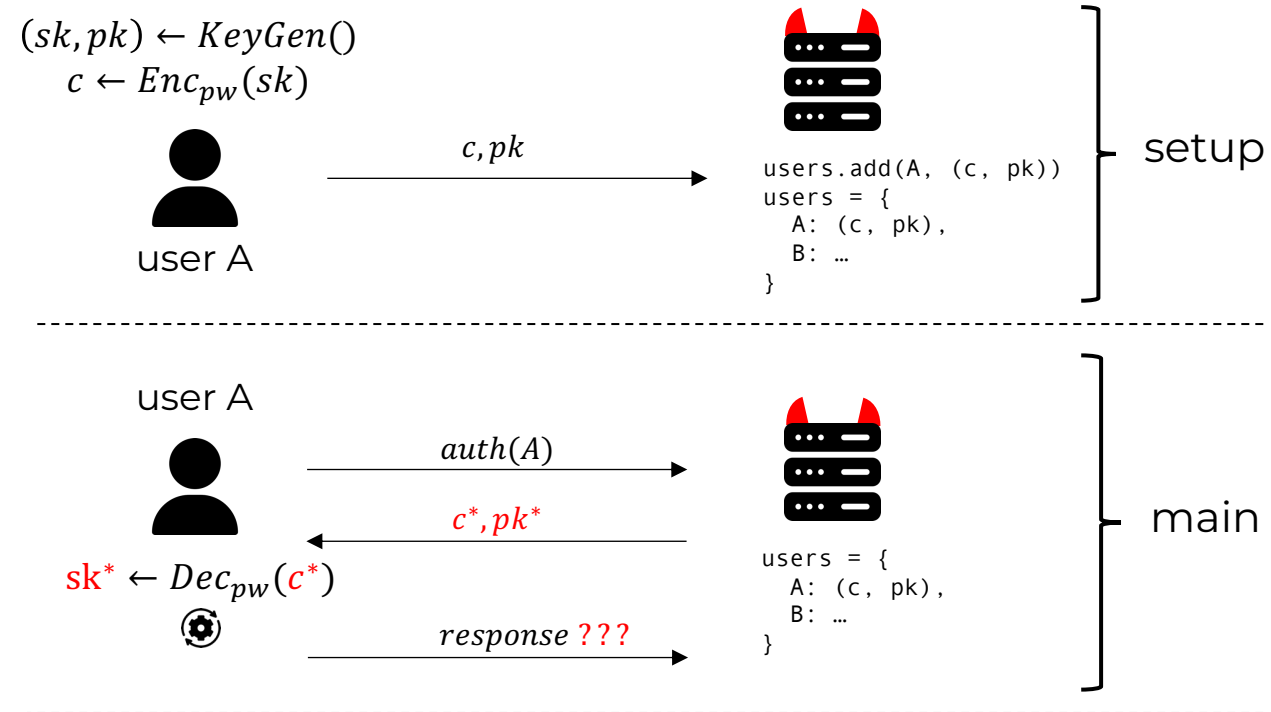
KO Attacks: Threat Model

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



KO Attacks: Goal and Variants

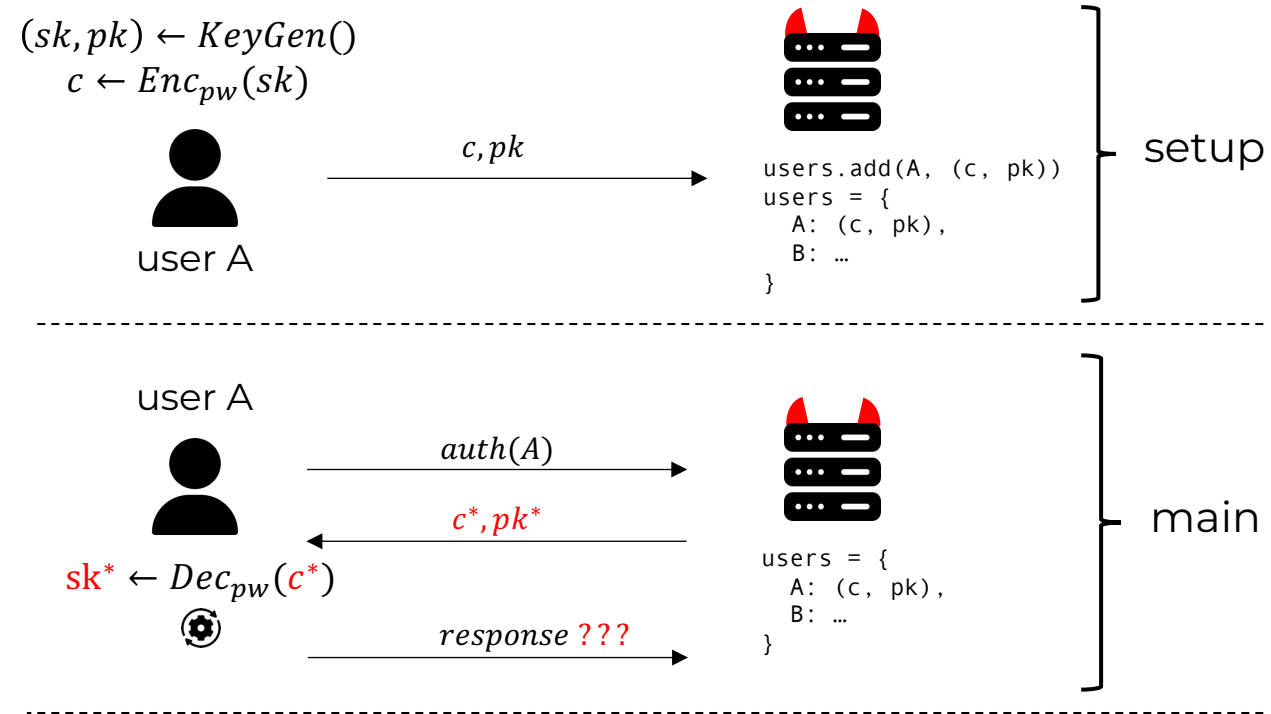
- Attacker goal:
 - Key recovery (of sk)
 - (or weaker goals like decryption)
- Overwriting control varies:
 - only pk ,
 - both c and pk ,
 - or controlled vs. blind overwriting?
- Efficient in practice
 - Minimize # of main phases



Key Overwriting Attacks

- Questions?
 - On the setting (setup & main phases)
 - On the threat model (malicious server)
 - ...

- Up next: KO attack on OpenPGP



KO Attack on OpenPGP



OpenPGP

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



src: <https://xkcd.com/1181/>

Refresher: Digital Signature Algorithm

- 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 $H: \{0,1\}^* \rightarrow \mathbb{Z}_q$
 - public key $pk = (p, q, g, y := g^x \text{ mod } p)$
- Operations

Takeaway:
infeasible to compute
secret (DLOG)

Takeaway:
public parameters
fixed by signer

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

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

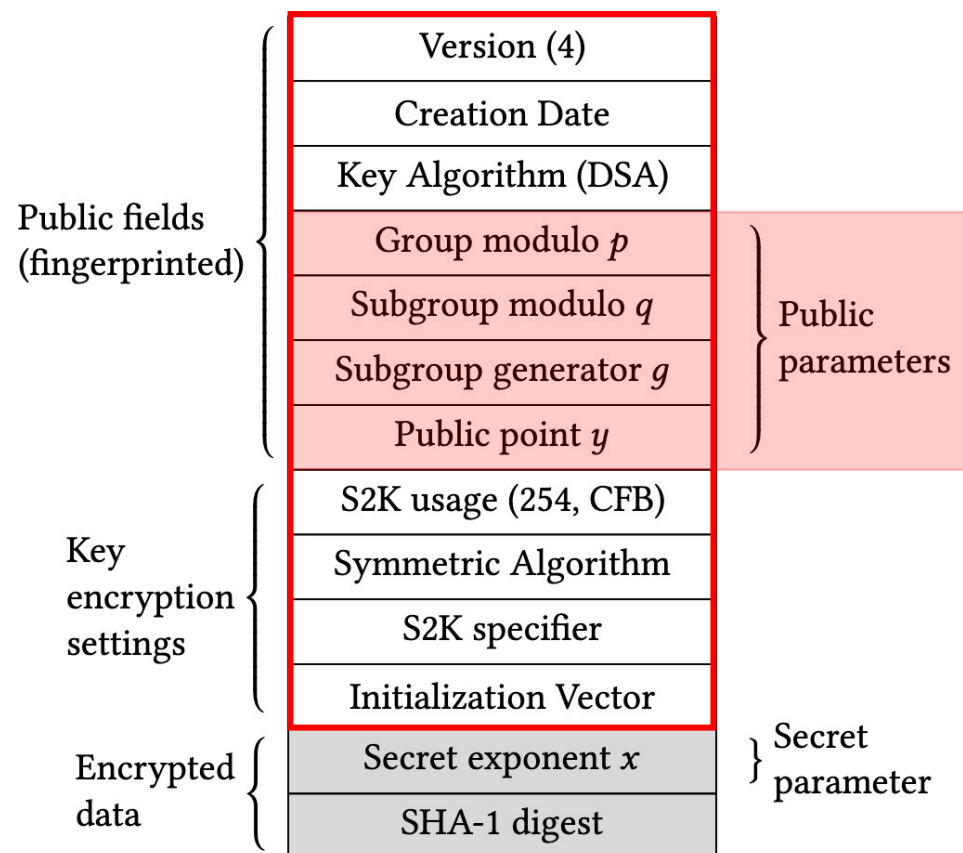
OpenPGP DSA Key Format

- No cryptographic protection for white fields
- Confidentiality guarantees for gray fields

Takeaway:

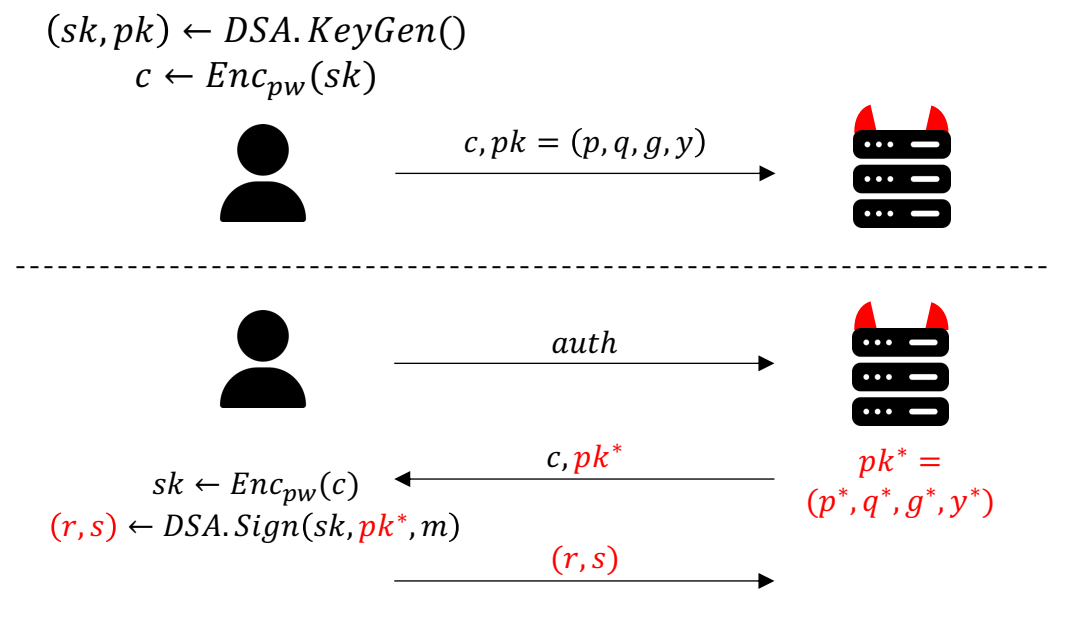
OpenPGP allows server to overwrite DSA public params

OpenPGP format:



Klíma and Rosa Attack on OpenPGP [1]

- Key overwriting
 - p^* prime such that $p^* - 1$ is smooth
 - g^* generator of $(\mathbb{Z}_{p^*})^*$
 - $q^* > p^*$ (normally, $ord(g^*) \leq p^* - 1$)
- Given (r, s) , recover nonce 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 on OpenPGP [1]

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

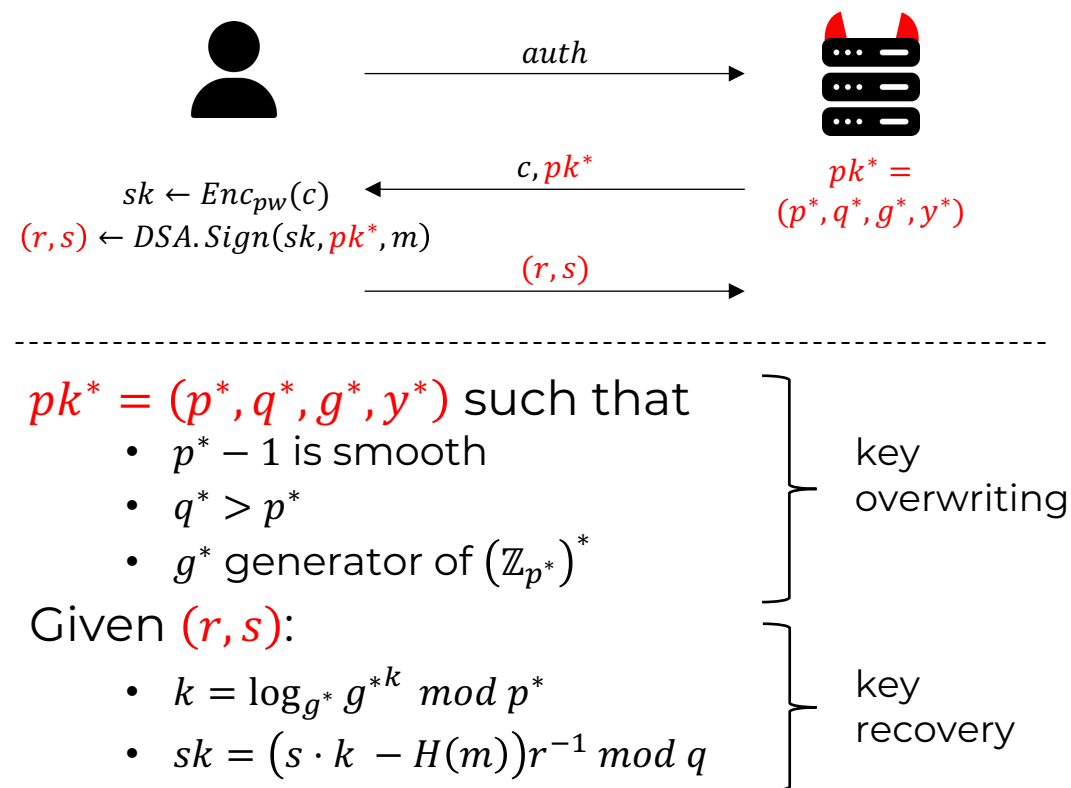
Takeaway:

Server can overwrite DSA public params with bogus ones that make DLOG easy, and recover secret key

[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)



[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 $x_i = x \bmod q_i$ (whp):
 - for $x_i = 0, 1, \dots, q_i - 1$:
 - $pk = (p^*, q_i, g^*, y = (g^*)^{x_i} \bmod p)$
 - if $Verify(pk, m, (r, s))$: return x_i
- Recover x with CRT from enough samples $x_i = x \bmod q_i$ for $i = 1, \dots, n$

```
Verify(pk, m, (r, s)):
  u1 = H(m)s-1 mod q
  u2 = rs-1 mod q
  return r = gu1yu2 mod p mod q
```

```
Sign(sk = x, pk = (p, q, g, y), m):
  k ←$ ℤq
  r = gk 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 $x_i = x \bmod q_i$ (whp):
 - for $x_i = 0, 1, \dots, q_i - 1$:
 - $pk = (p^*, q_i, g^*, y = (g^*)^{x_i} \bmod p)$
 - if $Verify(pk, m, (r, s))$: return x_i
- Recover x with CRT from enough samples $x_i = x \bmod q_i$ for $i = 1, \dots, n$

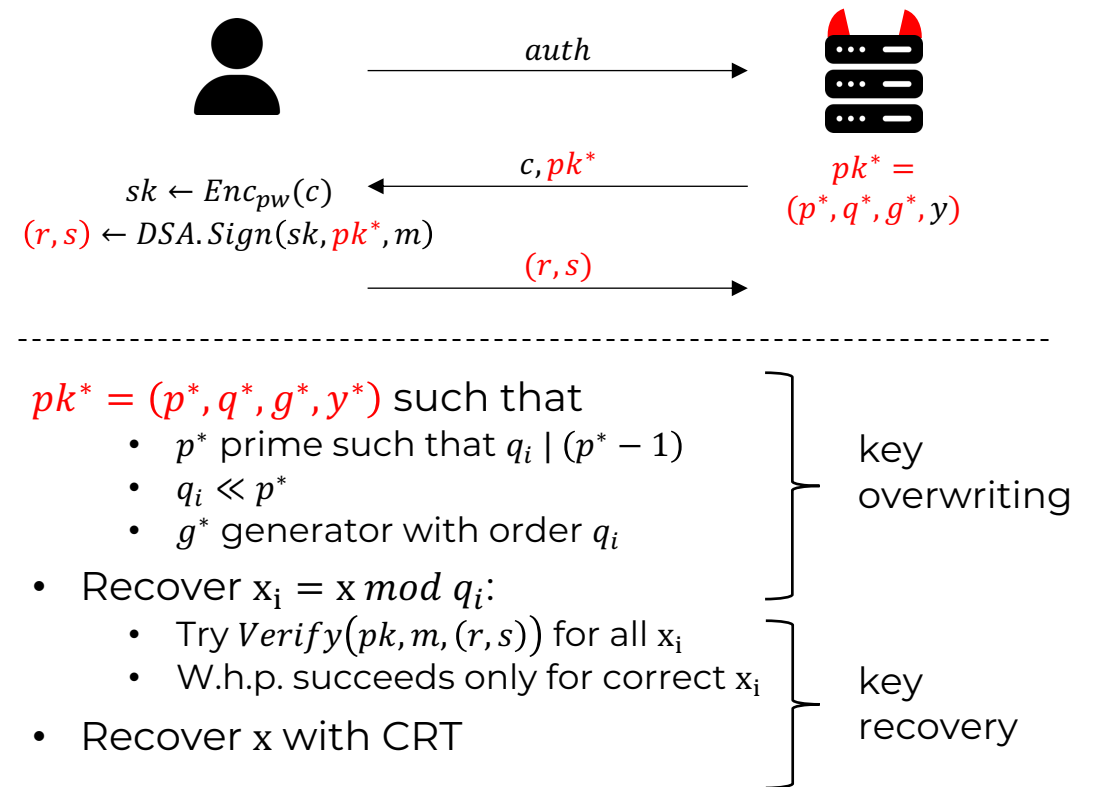
Takeaway:

Use small instead of large q primes to recover fractions of the secret key, combine them to the full secret

[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 $n \approx 16$ interactions (for 256-bit sk)
- Mitigation:
 - Small prime check insufficient
- Much more in paper:
 - Cross-algorithm attack
 - Attack exploiting key validation (KOKV-KR)
 - More attacks



[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 the Klíma and Rosa attack?
 - On the Victory by KO attacks?

- Up next: attacks on MEGA

$pk^* = (p^*, q^*, g^*, y^*)$ such that

- $p^* - 1$ is smooth
- $q^* > p^*$
- g^* generator of $(\mathbb{Z}_{p^*})^*$

Given (r, s) :

- $k = \log_{g^*} g^{*k} \bmod p^*$
- $sk = (s \cdot k - H(m))r^{-1} \bmod q$

key
overwriting

key
recovery

Klíma and Rosa

$pk^* = (p^*, q^*, g^*, y^*)$ such that

- p^* prime such that $q_i \mid (p^* - 1)$
- $q_i \ll p^*$
- g^* generator with order q_i

• 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
overwriting

key
recovery

Victory by KO

Attacks on MEGA

A dramatic, black and white photograph of a stormy night sky. The sky is filled with dark, heavy clouds, and several bright, jagged lightning bolts are striking down from the clouds. The lightning bolts are the primary source of light, illuminating the surrounding clouds and the landscape below. The landscape is mostly in silhouette, showing a dark horizon line with some faint lights and structures. The overall mood is ominous and powerful.

MEGA: Setting

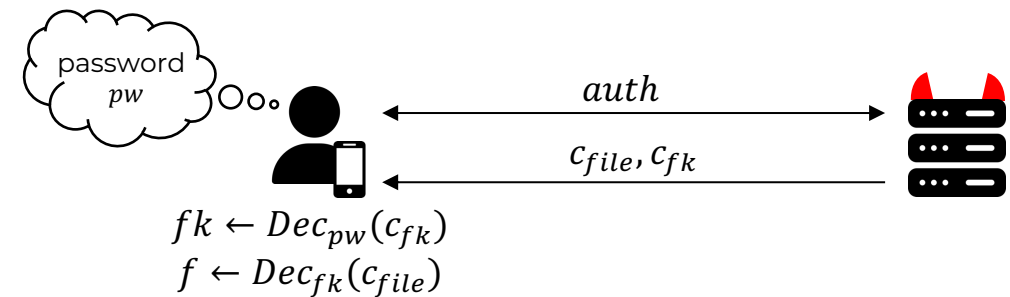
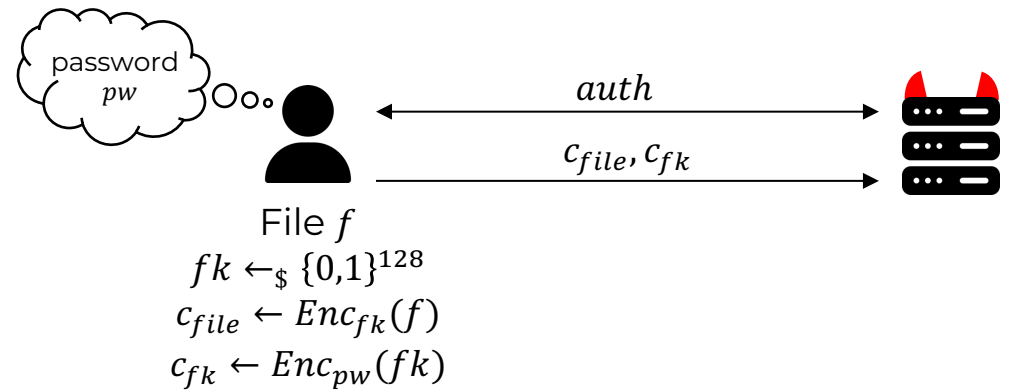
- The largest end-to-end encrypted cloud storage
 - 300M+ accounts storing 140B+ files [3]
 - Claiming strong privacy, even against themselves



[3] <https://mega.io/about> (03/2024)

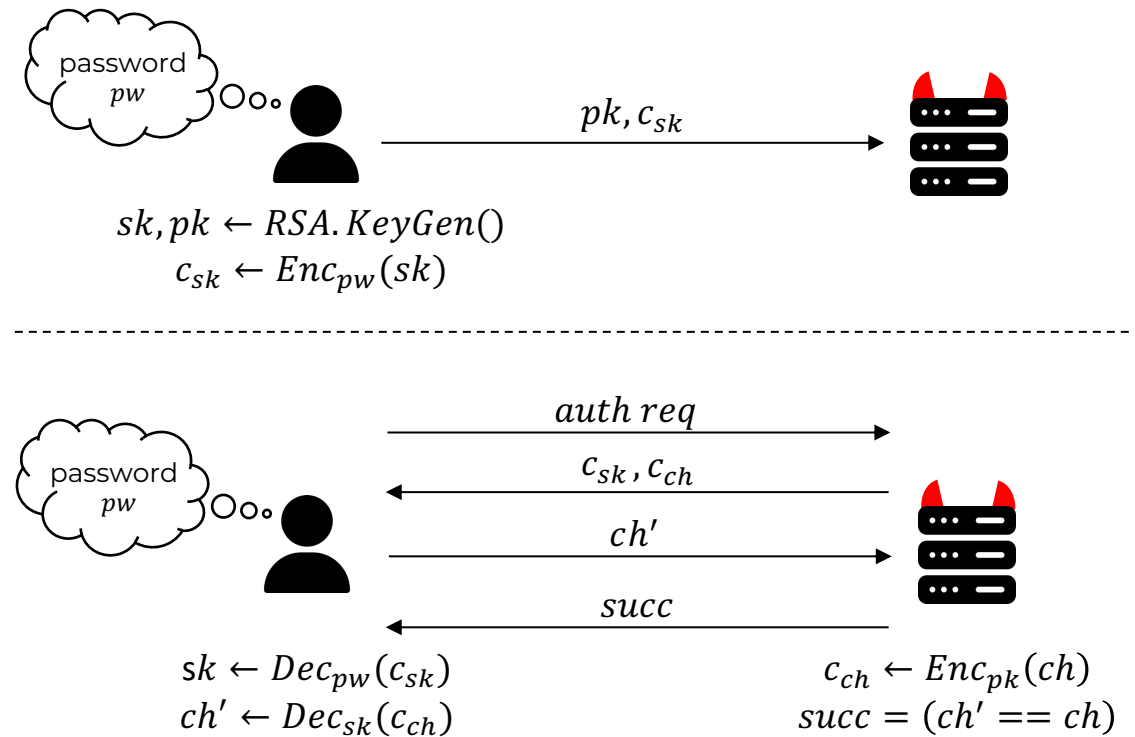
MEGA: Design-Files

- 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

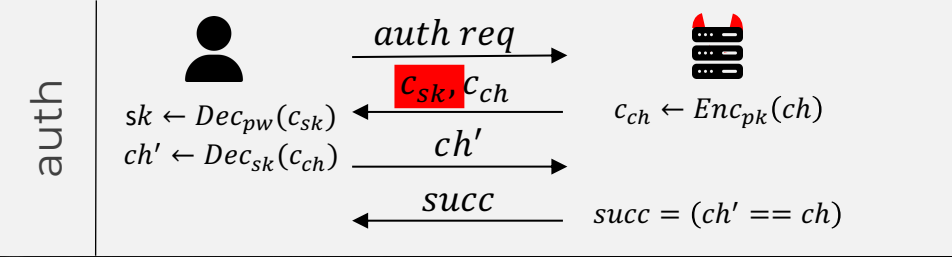


MEGA: Design–User Authentication

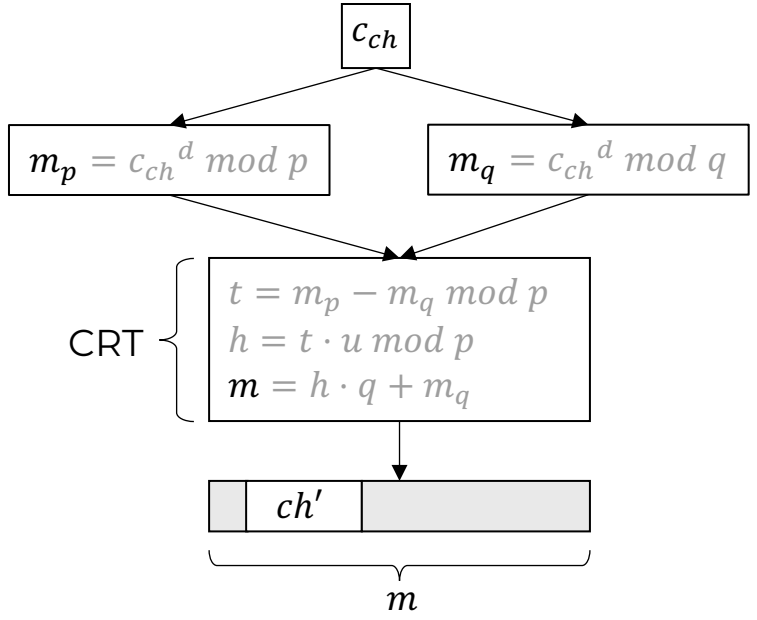
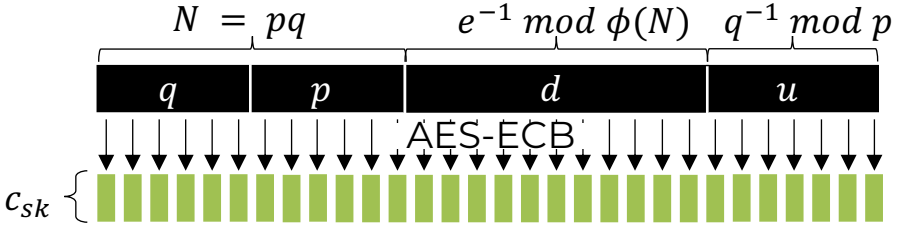
- Account registration
 - Generate RSA key pair (sk, pk)
 - Encrypt secret key sk with pw
 - Upload pk and ciphertext c_{sk}
- Authentication *auth*
 - Client requests to authenticate
 - Server sends secret key ctxt c_{sk}
 - Server encrypts challenge for pk
 - Client recovers RSA sk using pw and decrypts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



MEGA: Attack 1 [4]

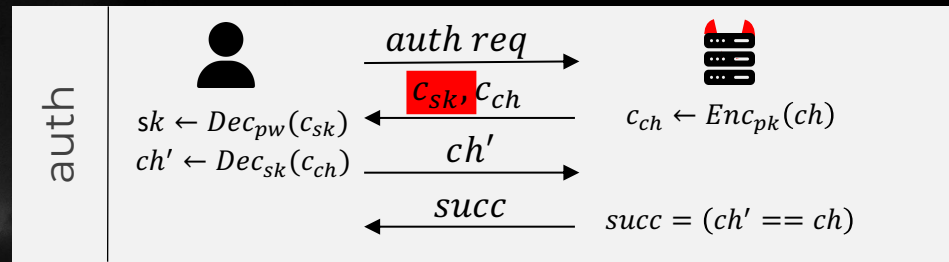


- RSA sk format
 - Primes p, q ; secret exponent d ;
 - u for RSA-CRT decryption
- RSA-CRT challenge decryption
 - Decrypt in \mathbb{Z}_p and \mathbb{Z}_q
 - Reconstruct $m \in \mathbb{Z}_N$ with CRT
 - Remove padding to recover ch'

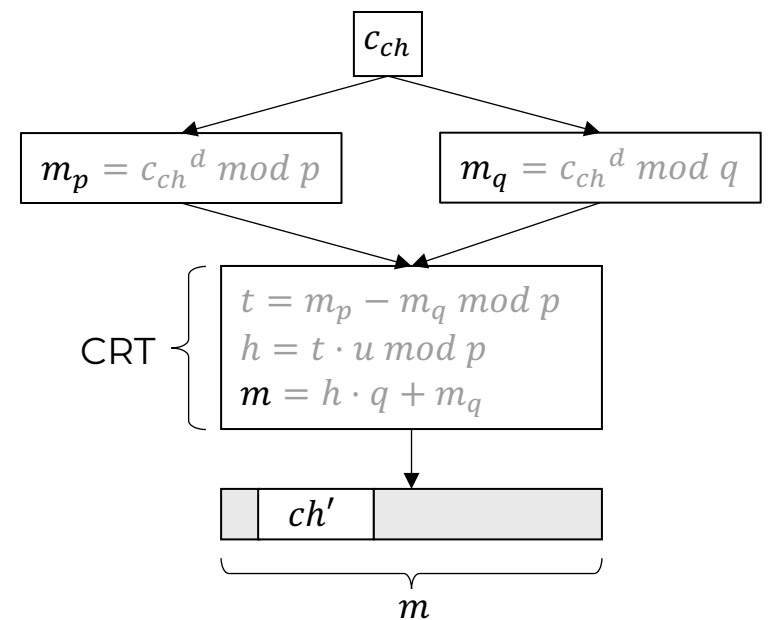
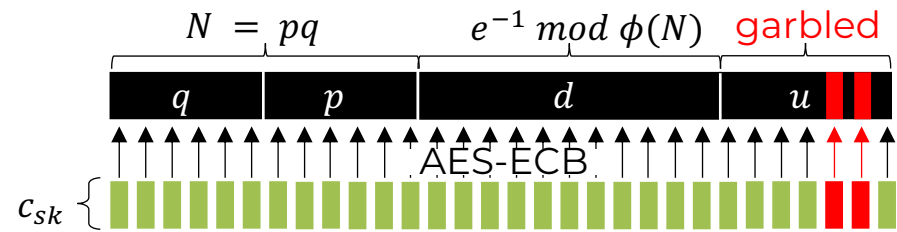


[4] Matilda Backendal, Miro Haller and Kenneth G. Paterson. "MEGA: Malleable Encryption Goes Awry". IEEE S&P 2023.

MEGA: Attack 1 [4]

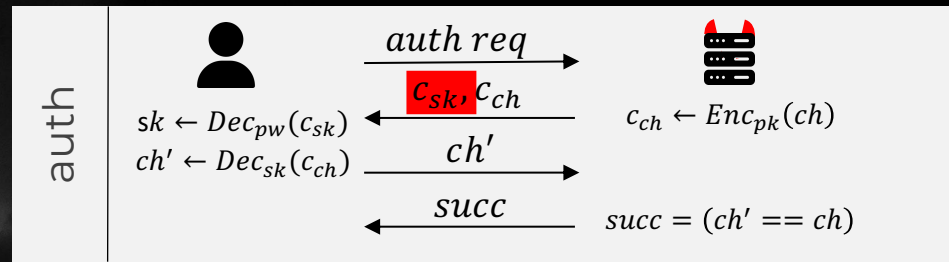


- Key overwriting
 - Overwriting some AES-ECB blocks garbles plaintext u
 - Without affecting $q, p, d!$

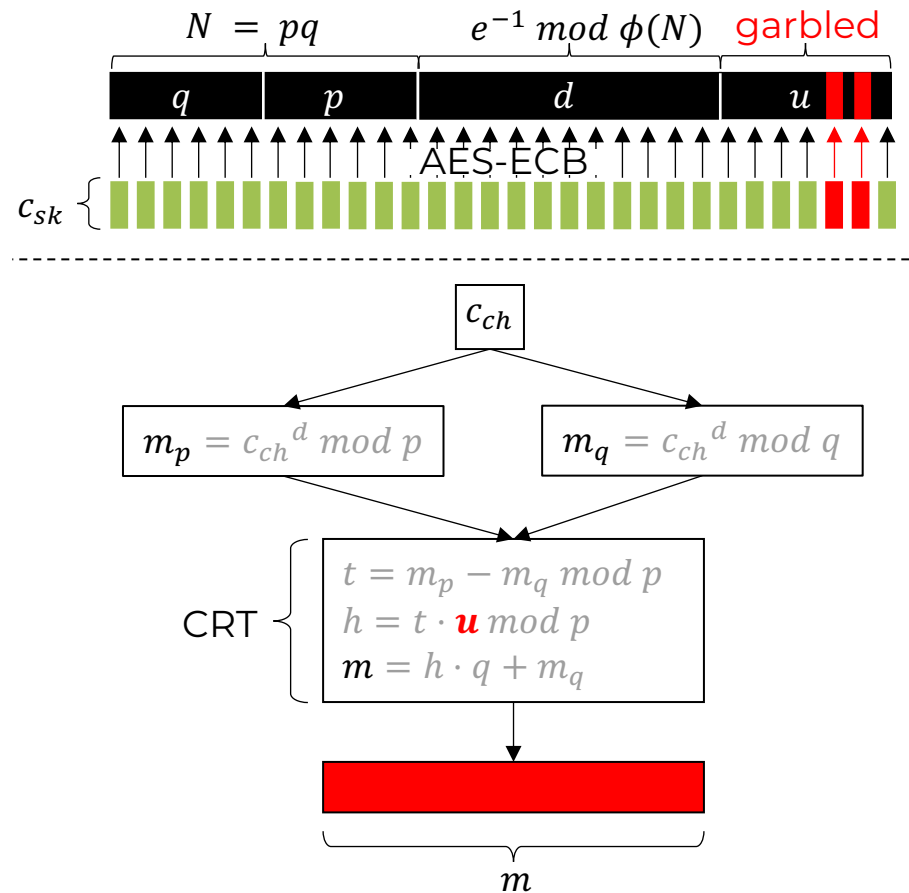


[4] Matilda Backendal, Miro Haller and Kenneth G. Paterson. "MEGA: Malleable Encryption Goes Awry". IEEE S&P 2023.

MEGA: Attack 1 [4]

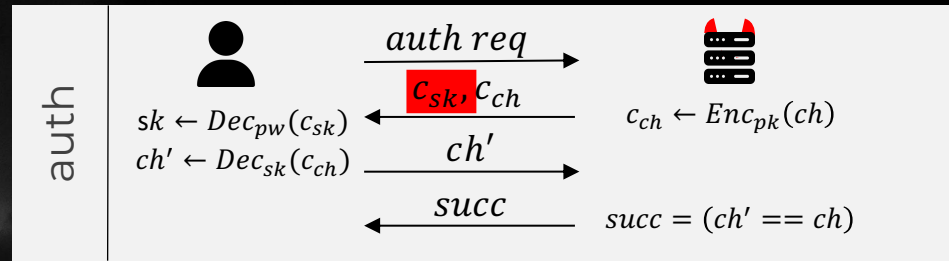


- Key overwriting
 - Overwriting some AES-ECB blocks garbles plaintext u
 - Without affecting $q, p, d!$
- In general, garbles decryption

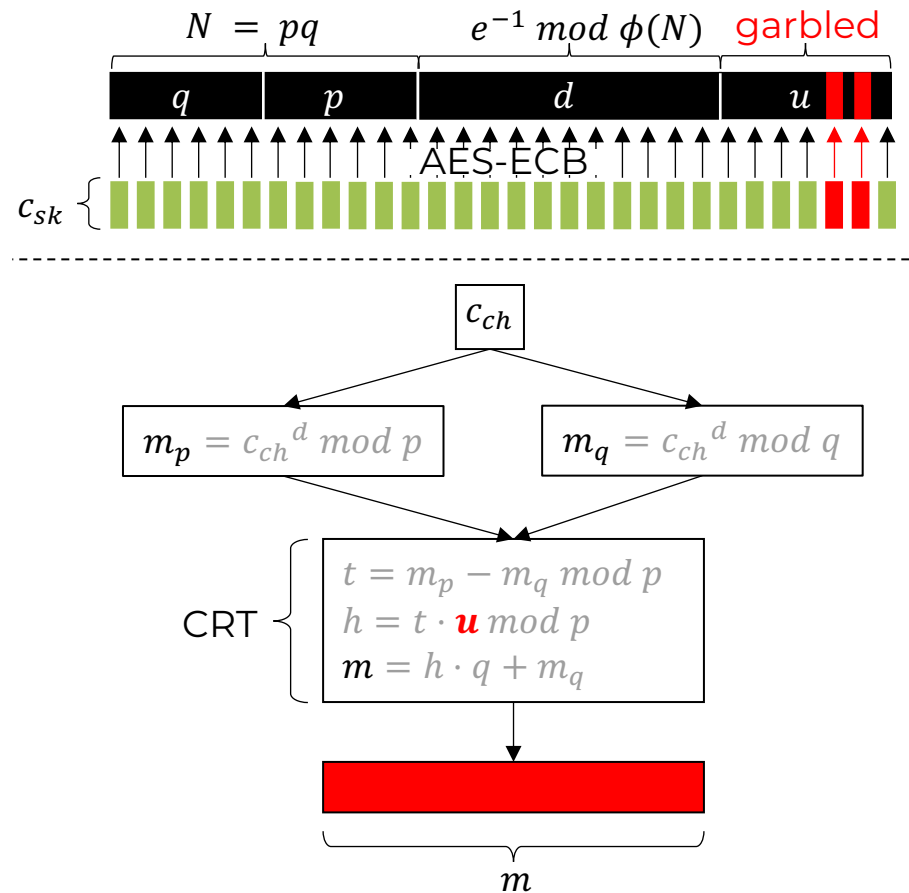


[4] Matilda Backendal, Miro Haller and Kenneth G. Paterson. "MEGA: Malleable Encryption Goes Awry". IEEE S&P 2023.

MEGA: Attack 1 [4]

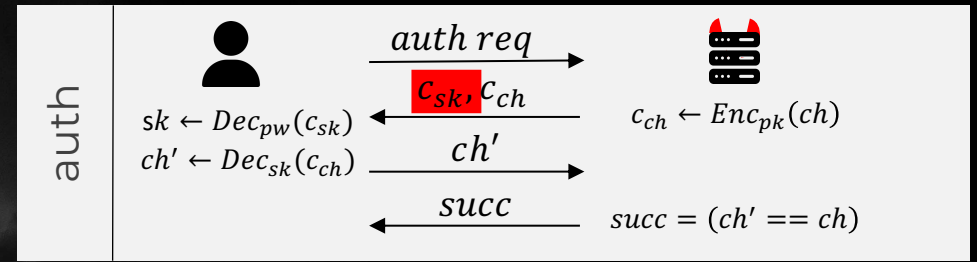


- Key overwriting
 - Overwriting some AES-ECB blocks garbles plaintext u
 - Without affecting $q, p, d!$
- In general, garbles decryption
- But still succeeds if $ch < p, q$

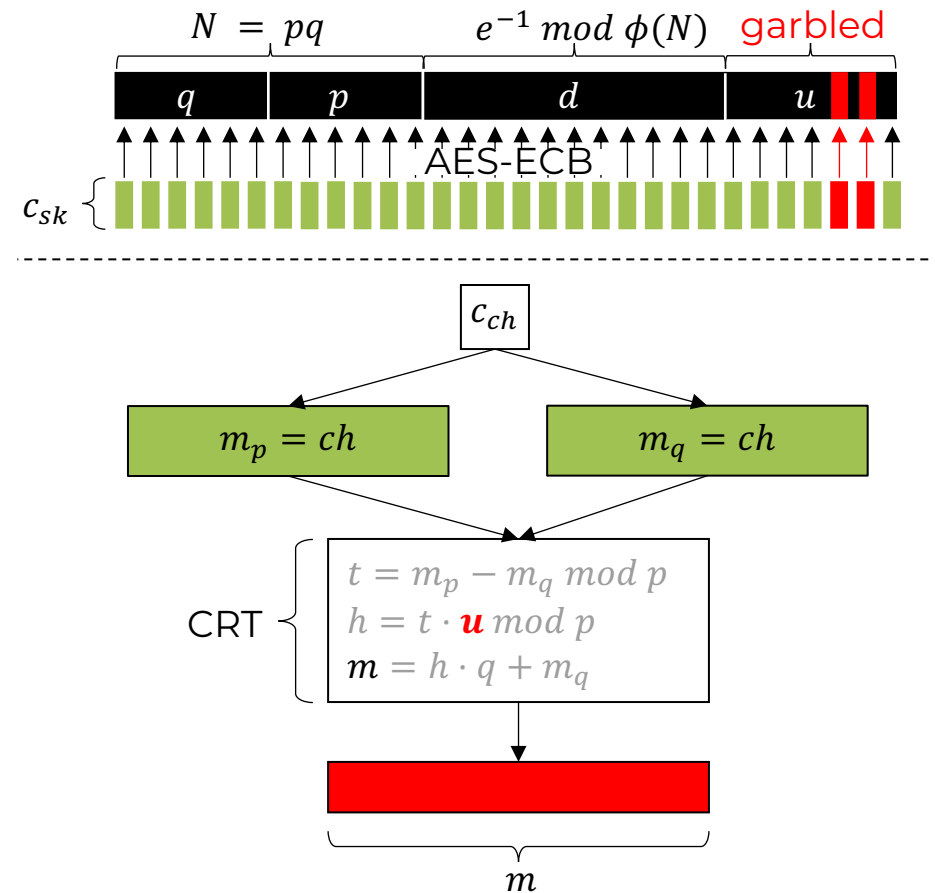


[4] Matilda Backendal, Miro Haller and Kenneth G. Paterson. "MEGA: Malleable Encryption Goes Awry". IEEE S&P 2023.

MEGA: Attack 1 [4]

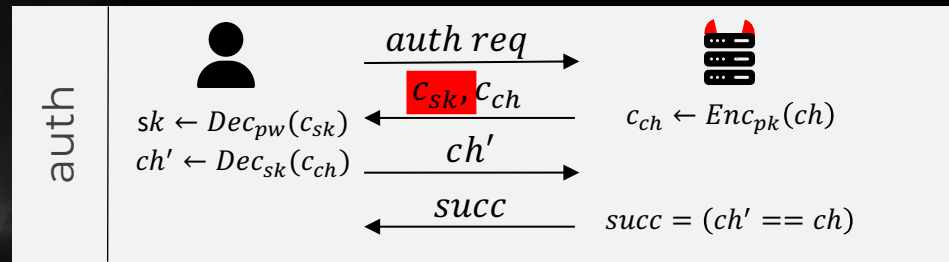


- Key overwriting
 - Overwriting some AES-ECB blocks garbles plaintext u
 - Without affecting $q, p, d!$
- In general, garbles decryption
- But still succeeds if $ch < p, q$
 - $m_p = m_q = ch$ because decryption in \mathbb{Z}_p and \mathbb{Z}_q is unique!

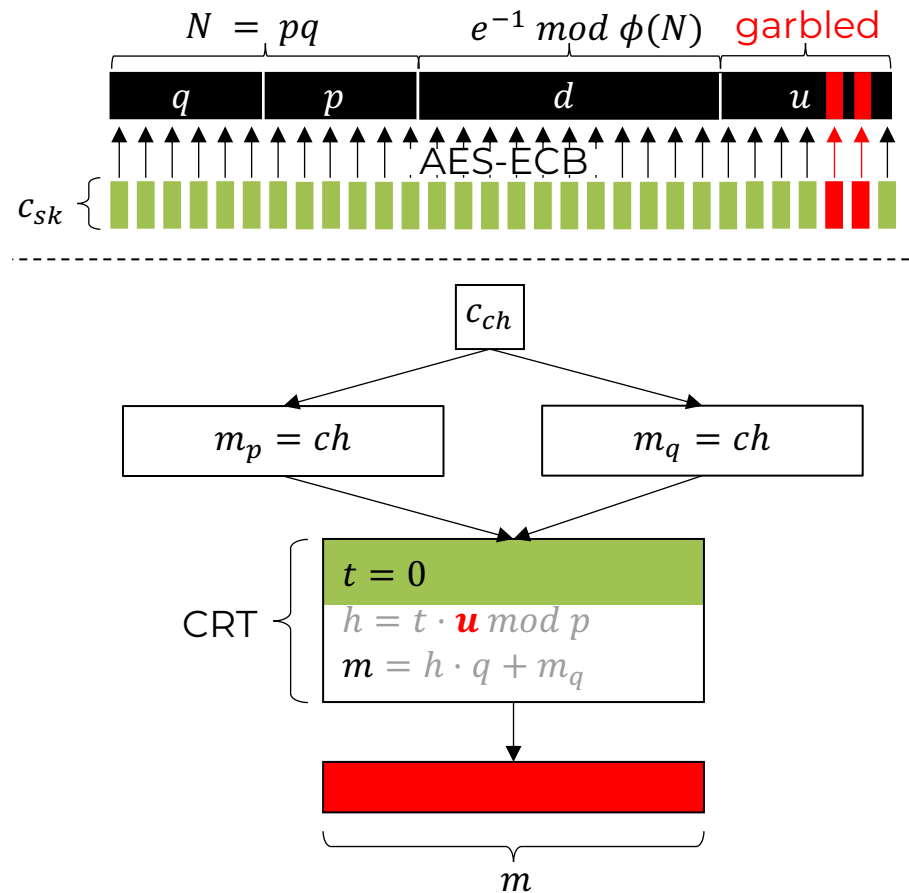


[4] Matilda Backendal, Miro Haller and Kenneth G. Paterson. "MEGA: Malleable Encryption Goes Awry". IEEE S&P 2023.

MEGA: Attack 1 [4]

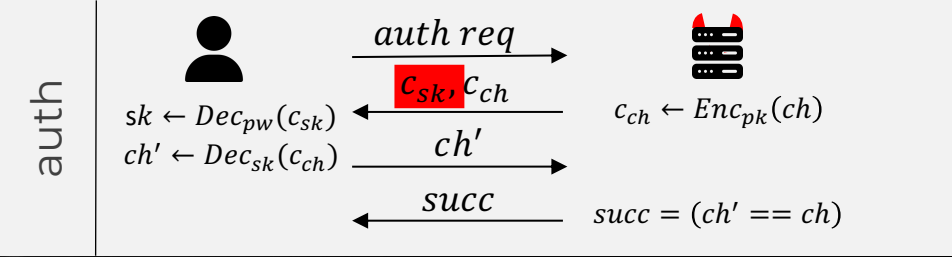


- Key overwriting
 - Overwriting some AES-ECB blocks garbles plaintext u
 - Without affecting $q, p, d!$
- In general, garbles decryption
- But still succeeds if $ch < p, q$
 - $m_p = m_q = ch$ because decryption in \mathbb{Z}_p and \mathbb{Z}_q is unique!
 - Simplifies CRT

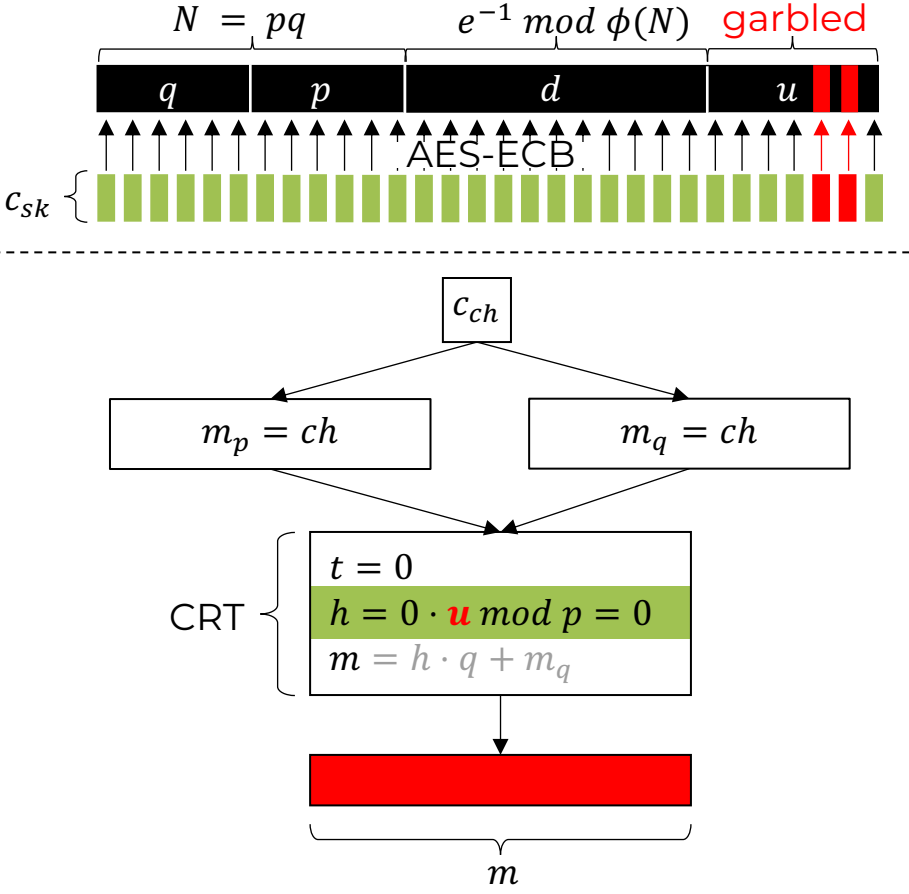


[4] Matilda Backendal, Miro Haller and Kenneth G. Paterson. "MEGA: Malleable Encryption Goes Awry". IEEE S&P 2023.

MEGA: Attack 1 [4]

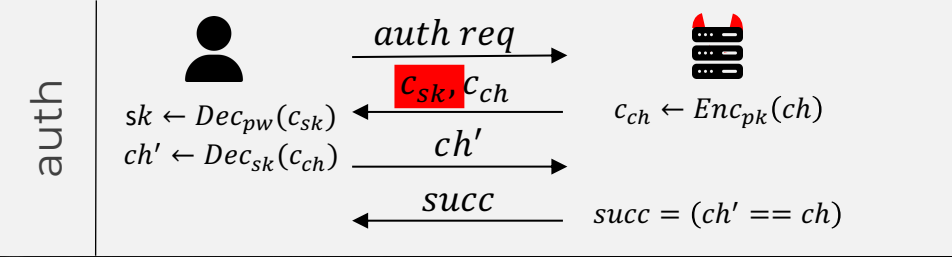


- Key overwriting
 - Overwriting some AES-ECB blocks garbles plaintext u
 - Without affecting $q, p, d!$
- In general, garbles decryption
- But still succeeds if $ch < p, q$
 - $m_p = m_q = ch$ because decryption in \mathbb{Z}_p and \mathbb{Z}_q is unique!
 - Simplifies CRT

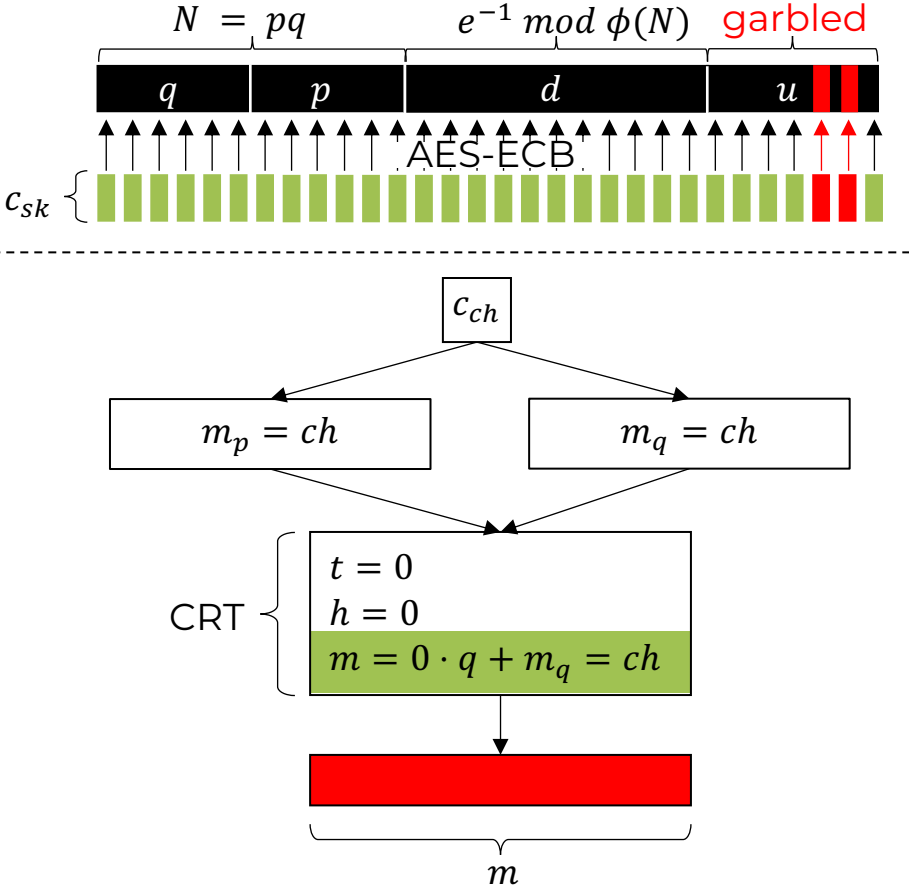


[4] Matilda Backendal, Miro Haller and Kenneth G. Paterson. "MEGA: Malleable Encryption Goes Awry". IEEE S&P 2023.

MEGA: Attack 1 [4]

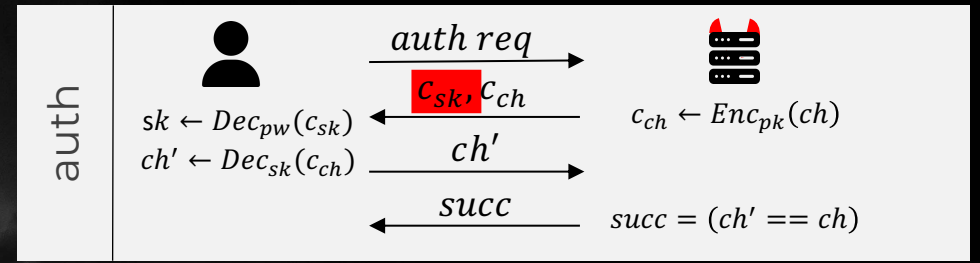


- Key overwriting
 - Overwriting some AES-ECB blocks garbles plaintext u
 - Without affecting $q, p, d!$
- In general, garbles decryption
- But still succeeds if $ch < p, q$
 - $m_p = m_q = ch$ because decryption in \mathbb{Z}_p and \mathbb{Z}_q is unique!
 - Simplifies CRT

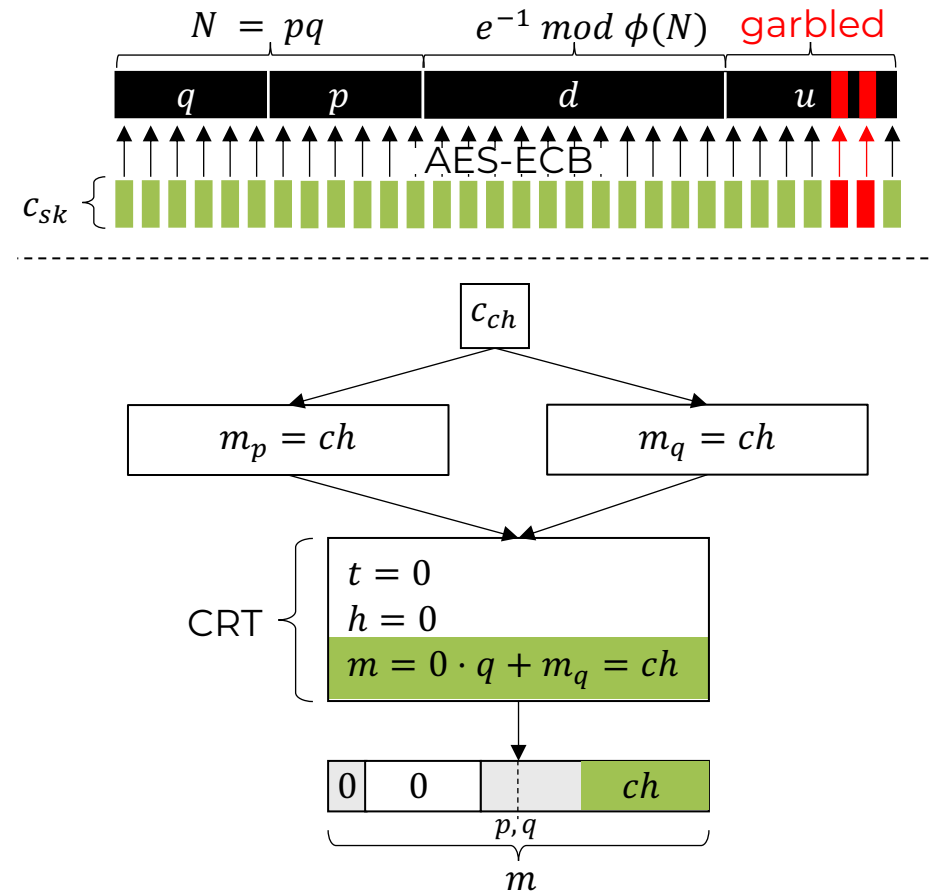


[4] Matilda Backendal, Miro Haller and Kenneth G. Paterson. "MEGA: Malleable Encryption Goes Awry". IEEE S&P 2023.

MEGA: Attack 1 [4]

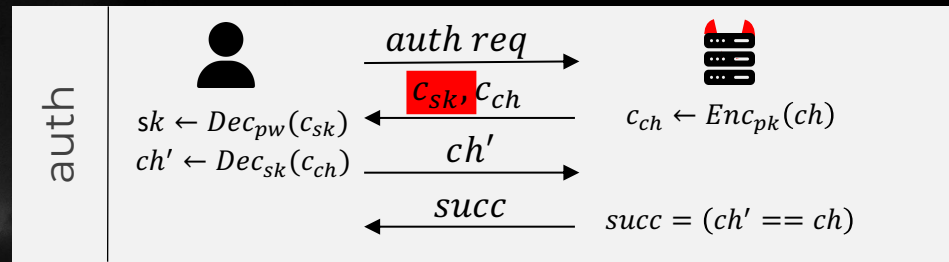


- Key overwriting
 - Overwriting some AES-ECB blocks garbles plaintext u
 - Without affecting $q, p, d!$
- In general, garbles decryption
- But still succeeds if $ch < p, q$
 - $m_p = m_q = ch$ because decryption in \mathbb{Z}_p and \mathbb{Z}_q is unique!
 - Simplifies CRT

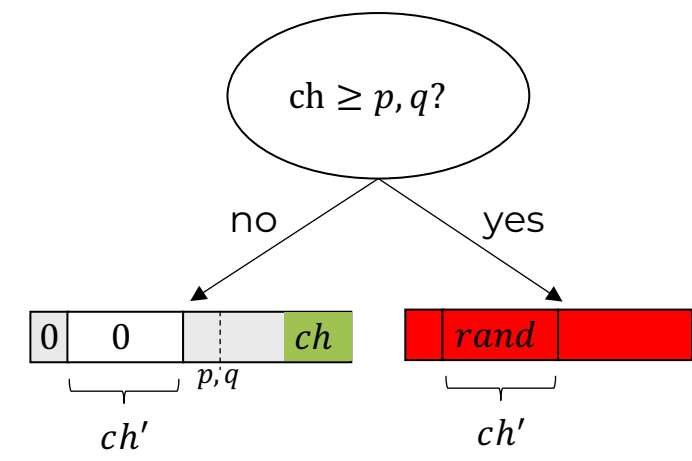
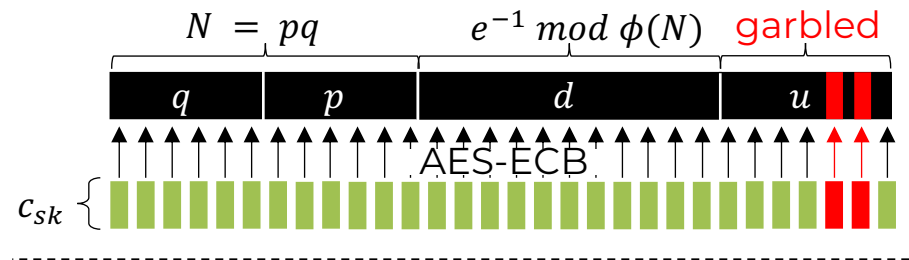


[4] Matilda Backendal, Miro Haller and Kenneth G. Paterson. "MEGA: Malleable Encryption Goes Awry". IEEE S&P 2023.

MEGA: Attack 1 [4]



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



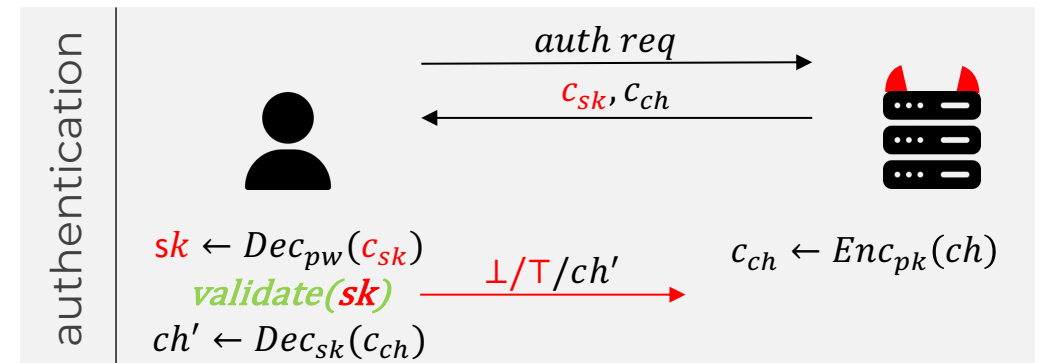
[4] Matilda Backendal, Miro Haller and Kenneth G. Paterson. "MEGA: Malleable Encryption Goes Awry". IEEE S&P 2023.

[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]

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

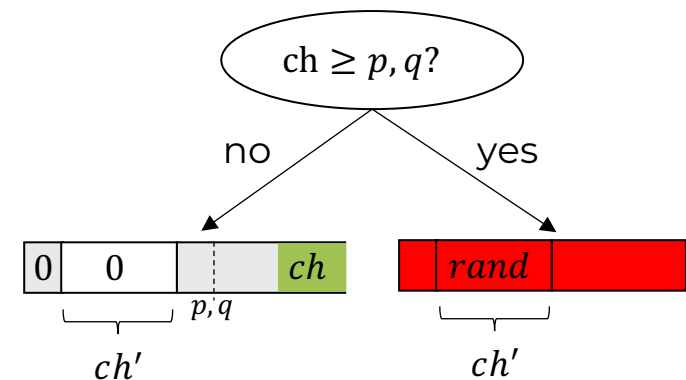
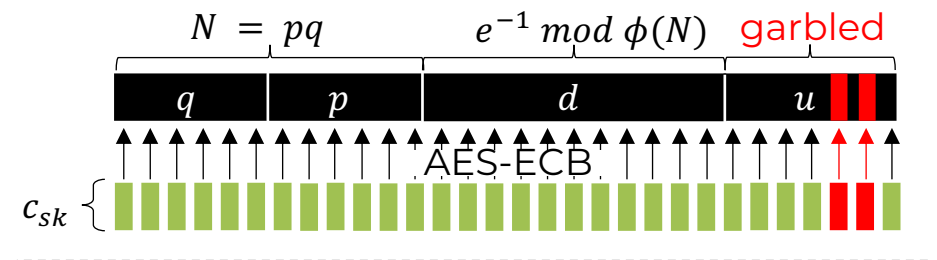
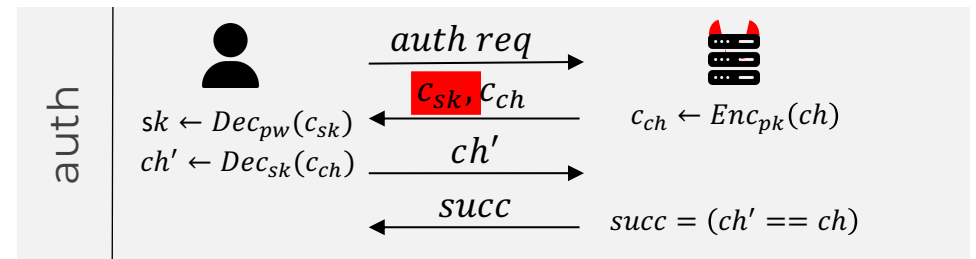


[6] Martin R. Albrecht, Miro Haller, Lenka Mareková, Kenneth G. Paterson. “Caveat Implementor! Key Recovery Attacks on MEGA”. Eurocrypt 2023.

MEGA Attacks: Questions?

- Questions?
 - On the MEGA key recovery attack?
 - On breaking the patches?

- Up next: root causes and mitigations

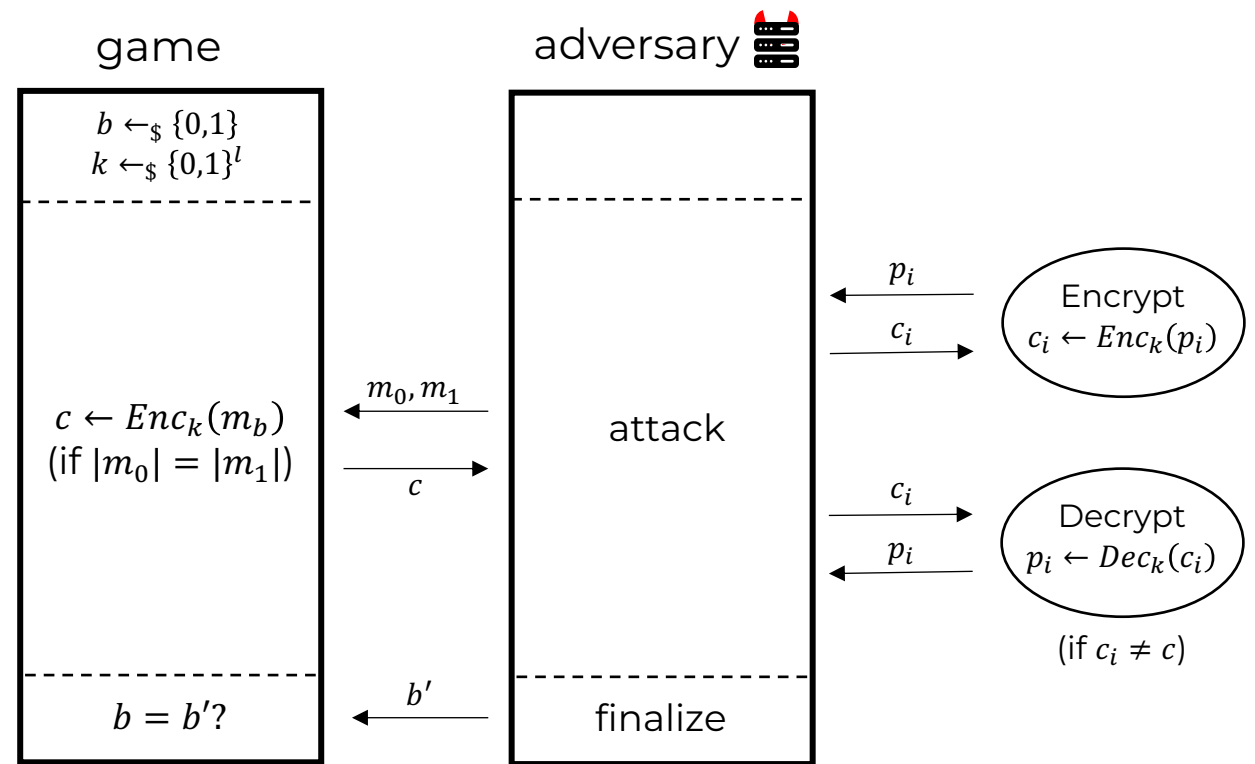


Root Causes and Mitigations



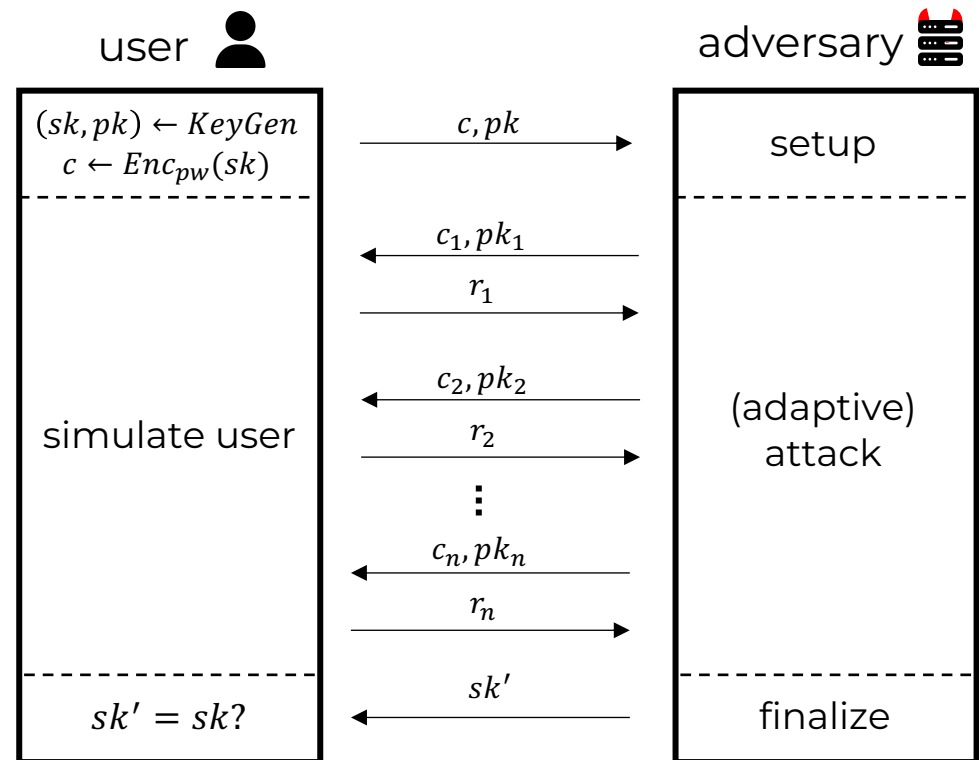
IND-CCA refresher

- Game picks random bit b and key k
- Adversary sends messages m_0, m_1
- Gets $c = Enc_k(m_b)$
- Enc and Dec oracles
- Guess b
- Security: low winning probability



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'
- Efficient if adversary has poly runtime and n is small

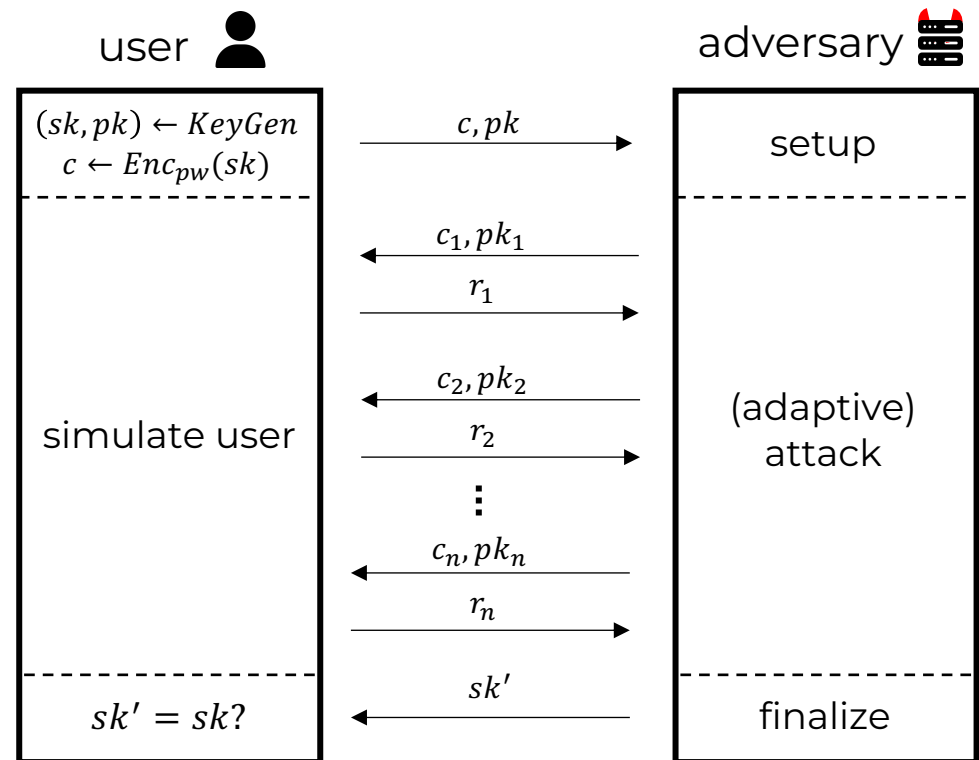


KO Key Recovery (KO-KR) Game

Lemma:
IND-CCA secure $\not\Rightarrow$ KO-KR secure

Proof sketch:

- Let Enc be IND-CCA secure, and $r = Dec_{pw}(c)$.
- Send $c = Enc_{pw}(sk)$, learn sk
- Win with prob. 1 if $sk' = sk$



KO Key Recovery (KO-KR) Game

Lemma:
IND-CCA secure $\not\Rightarrow$ KO-KR secure

Proof sketch:

- Let Enc be IND-CCA secure, and $r = Dec_{pw}(c)$.
- Send $c = Enc_{pw}(sk)$, learn sk
- Win with prob. 1 if $sk' = sk$

Takeaway:

The client-server interaction may have arbitrary leakage not captured by standard security notions.

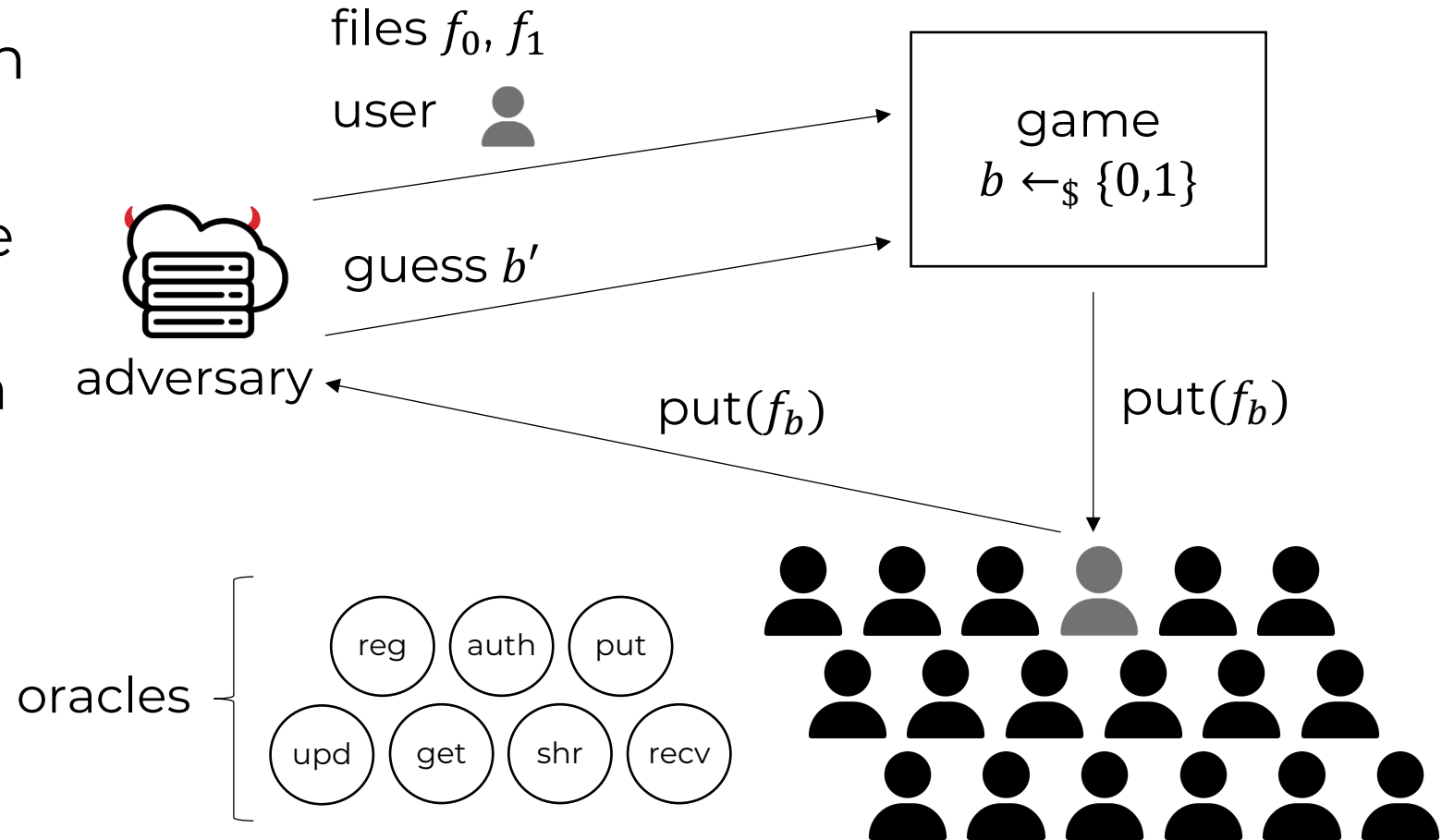
Security for E2EE Cloud Storage

- Identify core functionality
 - Register (reg)
 - Authenticate (auth)
 - Upload (put)
 - Update (upd)
 - Download (get)
 - Share (shr)
 - Receive (recv)
- Define syntax
 - Allowing non-atomic steps
 - Modeling arbitrary interleavings

Security for E2EE Cloud Storage

Security game intuition

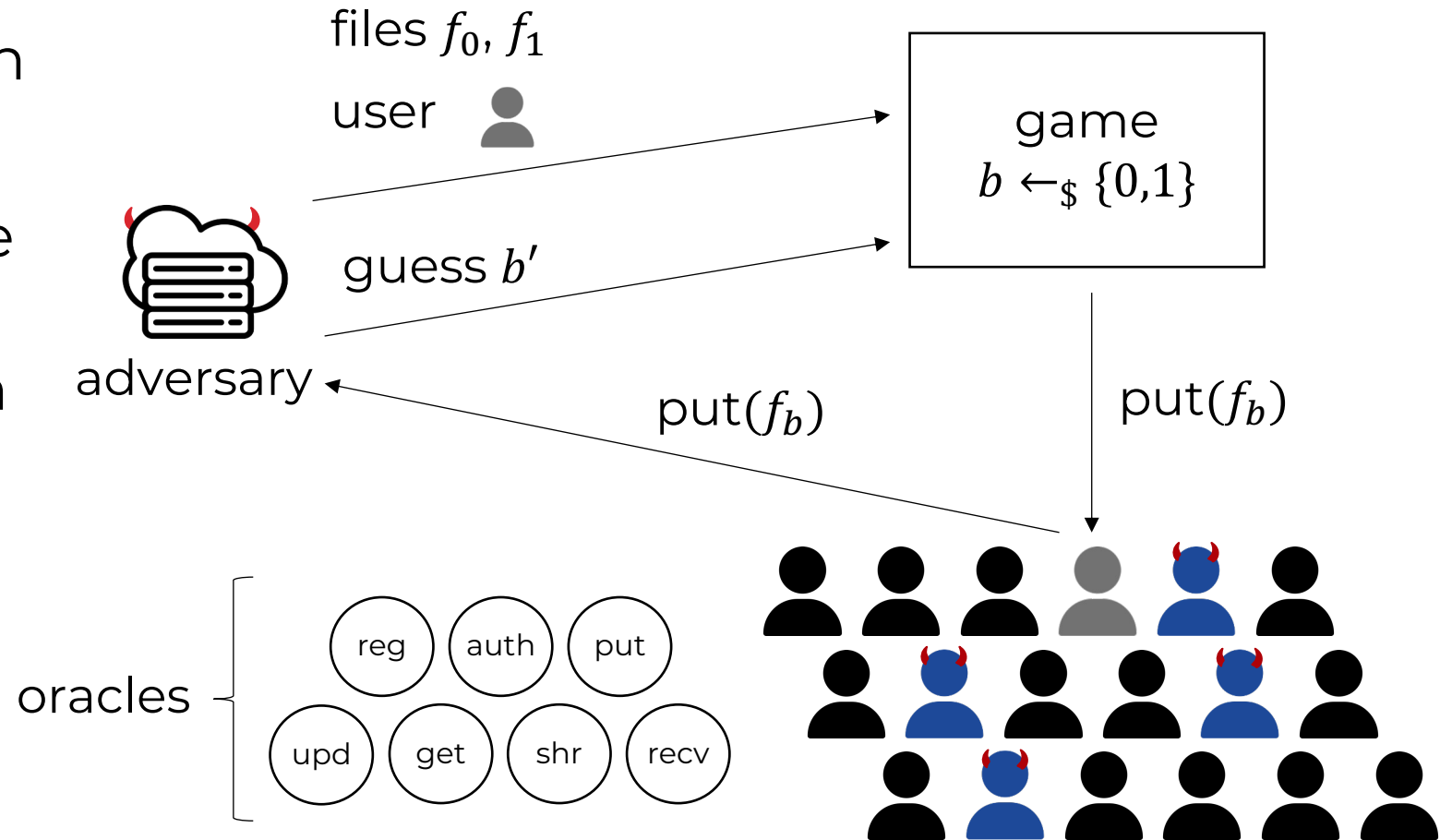
- Malicious server
- Full control over state
- Oracles to make honest users perform actions
- Provide two files f_0, f_1
- File f_b is uploaded
- Guess bit $b' = b$




Security for E2EE Cloud Storage

Security game intuition

- Malicious server
- Full control over state
- Oracles to make honest users perform actions
- Provide two files f_0, f_1
- File f_b is uploaded
- Guess bit $b' = b$
- User compromises

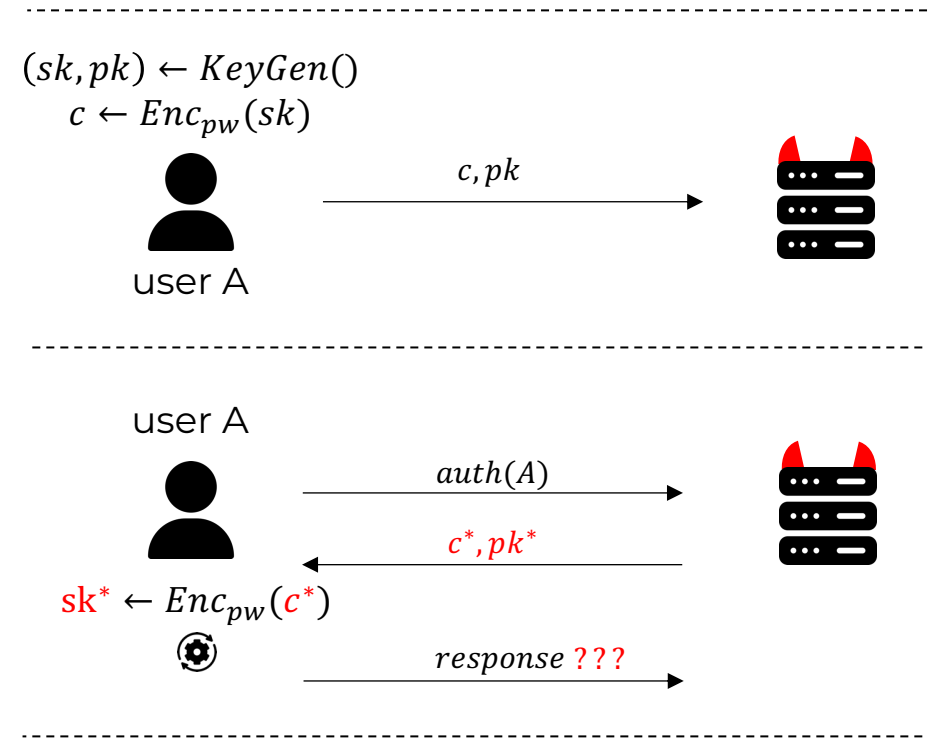


A dramatic, high-contrast black and white photograph of a stormy night. The sky is filled with dark, heavy clouds, and several bright, jagged lightning bolts are visible, striking down from the clouds. The foreground is mostly in silhouette, showing the dark outlines of trees and a distant horizon line. The overall mood is intense and powerful.

Summary: Key Overwriting Attacks

Key Overwriting Attacks

- E2EE services are more susceptible to KO attacks
 - OpenPGP: [[eprint:KR02](#)], [[CCS:BHP22](#)]
 - MEGA: [[IEEE SP:BHP23](#)], [[Eurocrypt:AHMP23](#)], [[PKC:RH23](#)]
- KO attack
 - malicious server
 - adaptively overwrites outsourced client key ciphertexts
 - observes client responses to recover key
- IND-CCA does not capture KO-KR
 - **custom security notions required for provable guarantees**



Backup Slides



Additional Resources

- Icons from flaticon.com:
 - [Freepik](#): user, server, systems update, cell phone