NTNU | Norwegian University of Science and Technology

MEGA: Malleable Encryption Goes Awry

Trial lecture, Tjerand Aga Silde, August 22nd, 2022

Content

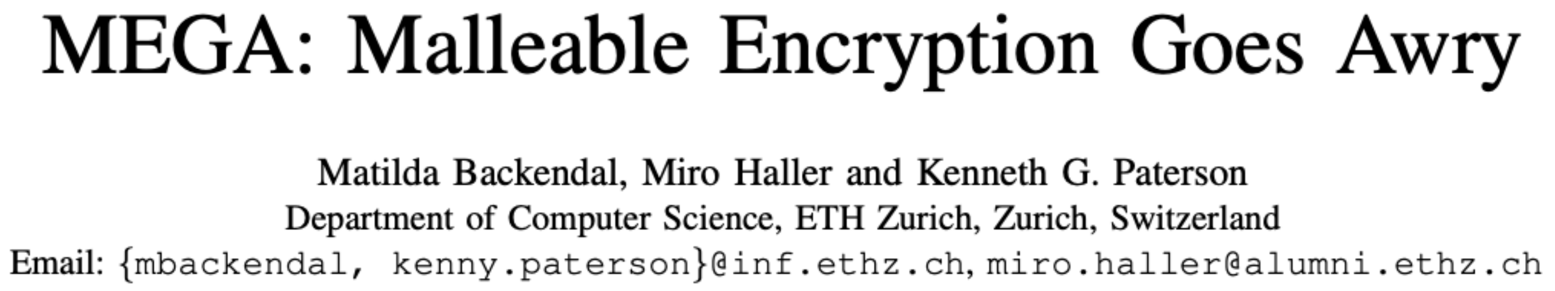
Cryptography: Theory vs Practice

- We design schemes and prove them secure w.r.t. some underlying hard problem, e.g. factoring or discrete logs.
- But proofs might have flaws. Or the underlying assumptions may not be as hard as we thought.
- There might be different versions of a scheme that are incompatible, or not even secure when tweaked.
- Implementations might not follow the specifications.
- Implementations might leak secret info while running.

Cryptography: Theory vs Practice

- The Merkle-Hellman Knapsack scheme was broken.
- Some instantiations over elliptic curves are not secure.
- PQC candidates Rainbow and SIKE was recently broken.
- Attacks on TLS: Bleichenbacher, POODLE, BEAST, CRIME, BREACH, Heartblead, Lucky13, LogJam,…
- OCB2 was standardized in 2009, but broken in 2019.
- Kocher broke schemes via side-channels in the 90's.
- The lack of integrity checks leads to decryption oracles.

Five Attacks on MEGA

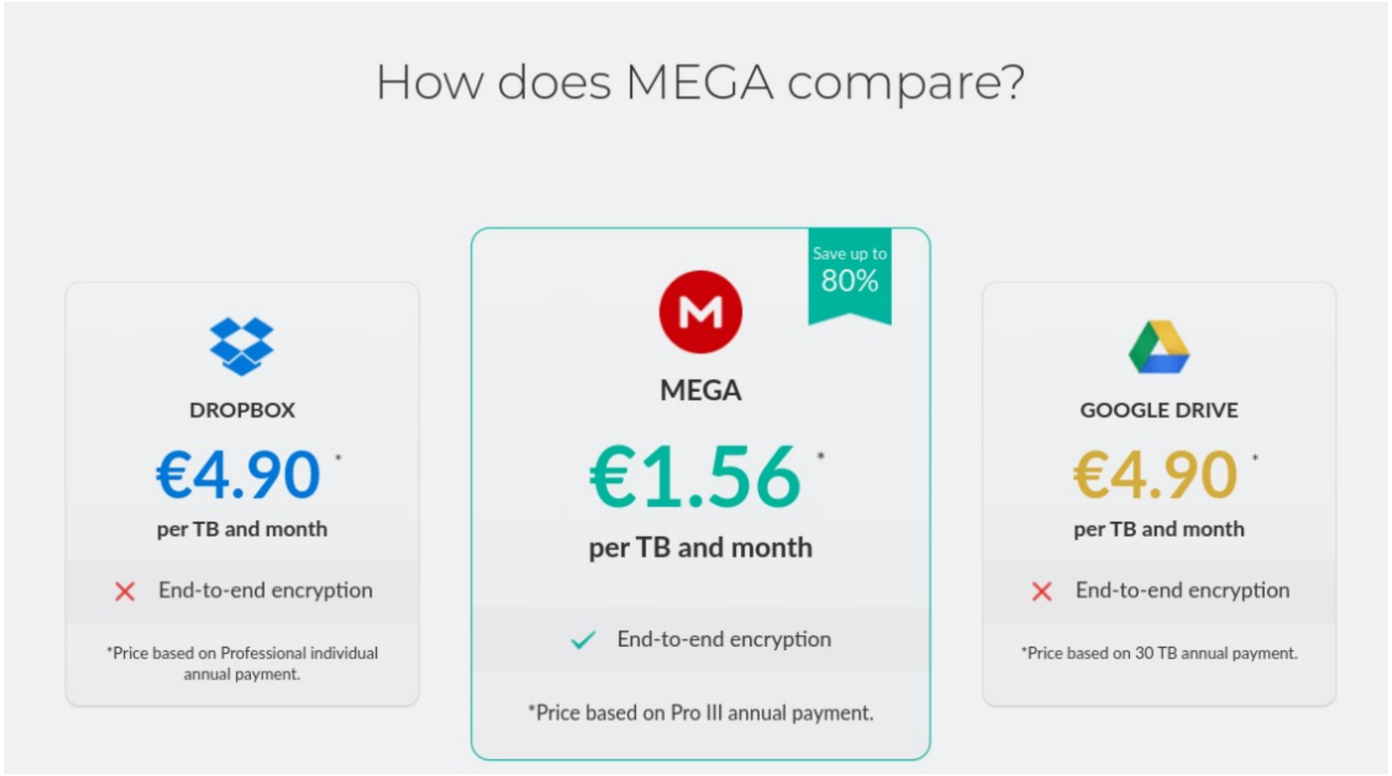


MEGA: Malleable Encryption Goes Awry

Matilda Backendal, Miro Haller and Kenneth G. Paterson
Department of Computer Science, ETH Zurich, Zurich, Switzerland
Email: {mbackendal, kenny.paterson}@inf.ethz.ch, miro.haller@alumni.ethz.ch

eprint.iacr.org/2022/959.pdf

Five Attacks on MEGA

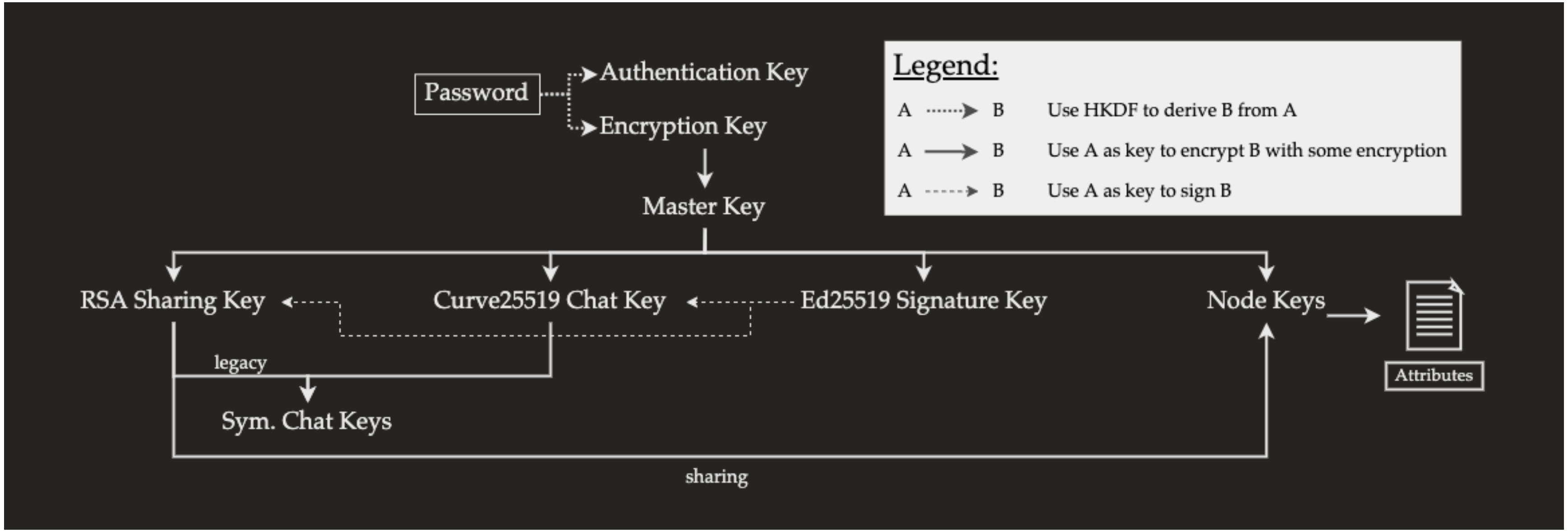


How does MEGA compare?

DROPBOX	MEGA	GOOGLE DRIVE
€4.90* per TB and month	€1.56* per TB and month	€4.90* per TB and month
✗ End-to-end encryption	✓ End-to-end encryption	✗ End-to-end encryption
*Price based on Professional individual annual payment.	*Price based on Pro III annual payment.	*Price based on 30 TB annual payment.

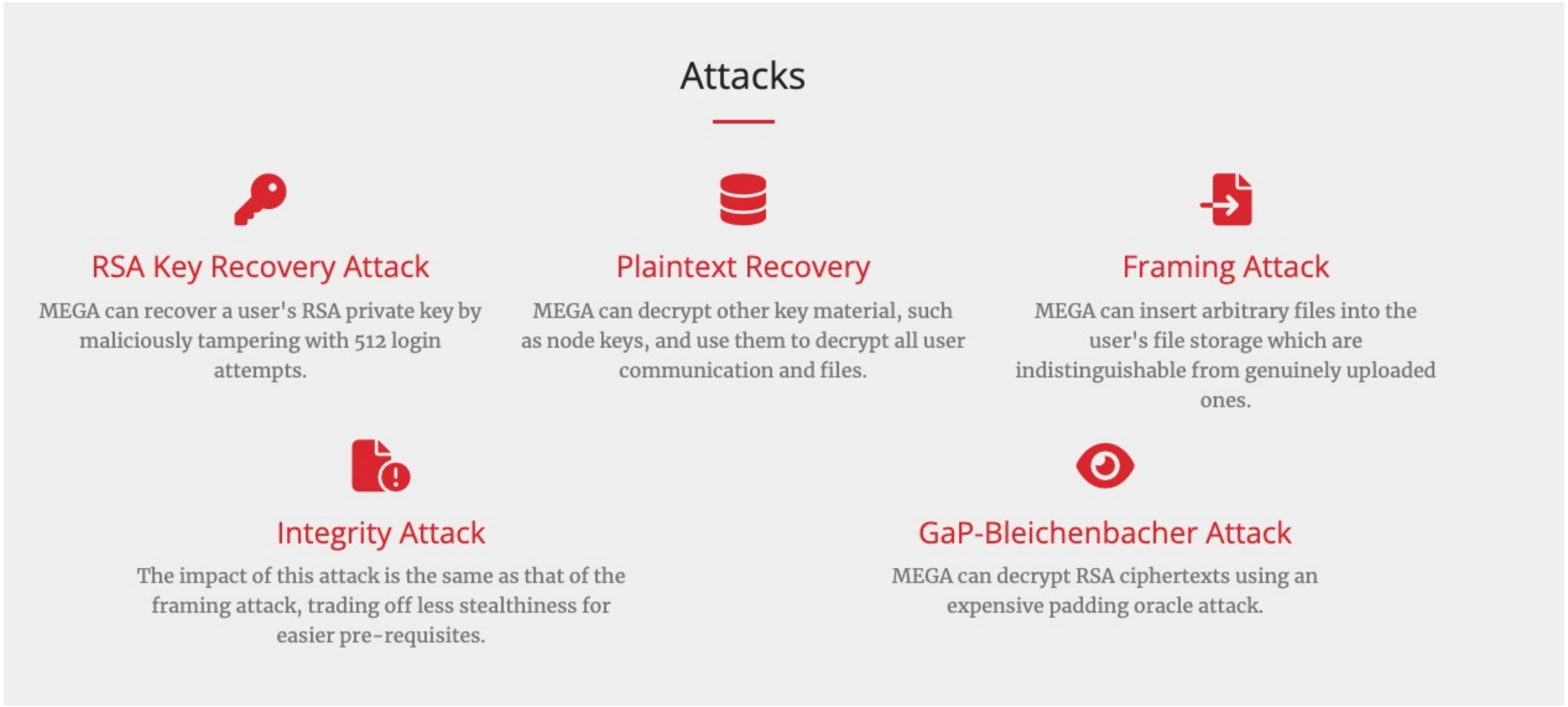
arstechnica.com/information-technology/2022/06/mega-says-it-cant-decrypt-your-files-new-poc-exploit-shows-otherwise

Five Attacks on MEGA



🚩 No integrity checks, AES-ECB, server-chosen plaintexts, custum padding, key-reuse 🚩

Five Attacks on MEGA



Attacks

RSA Key Recovery Attack

MEGA can recover a user's RSA private key by maliciously tampering with 512 login attempts.

Plaintext Recovery

MEGA can decrypt other key material, such as node keys, and use them to decrypt all user communication and files.

Framing Attack

MEGA can insert arbitrary files into the user's file storage which are indistinguishable from genuinely uploaded ones.

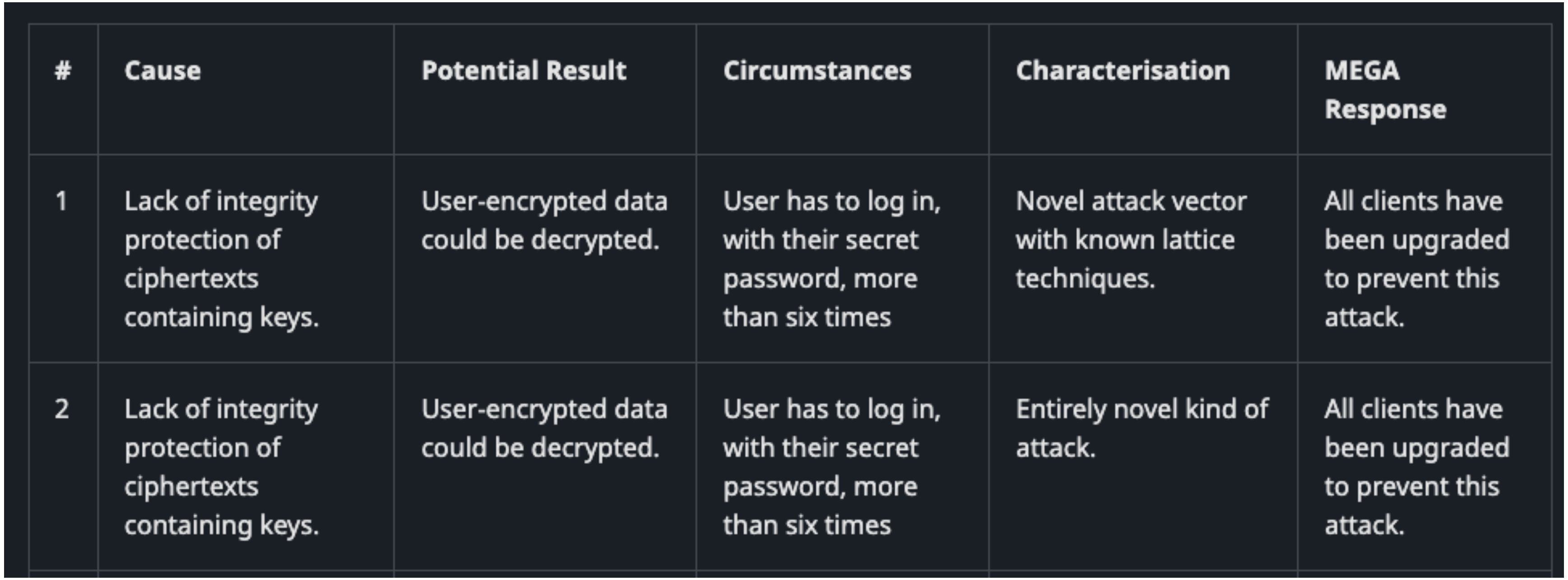
Integrity Attack

The impact of this attack is the same as that of the framing attack, trading off less stealthiness for easier pre-requisites.

GaP-Bleichenbacher Attack

MEGA can decrypt RSA ciphertexts using an expensive padding oracle attack.

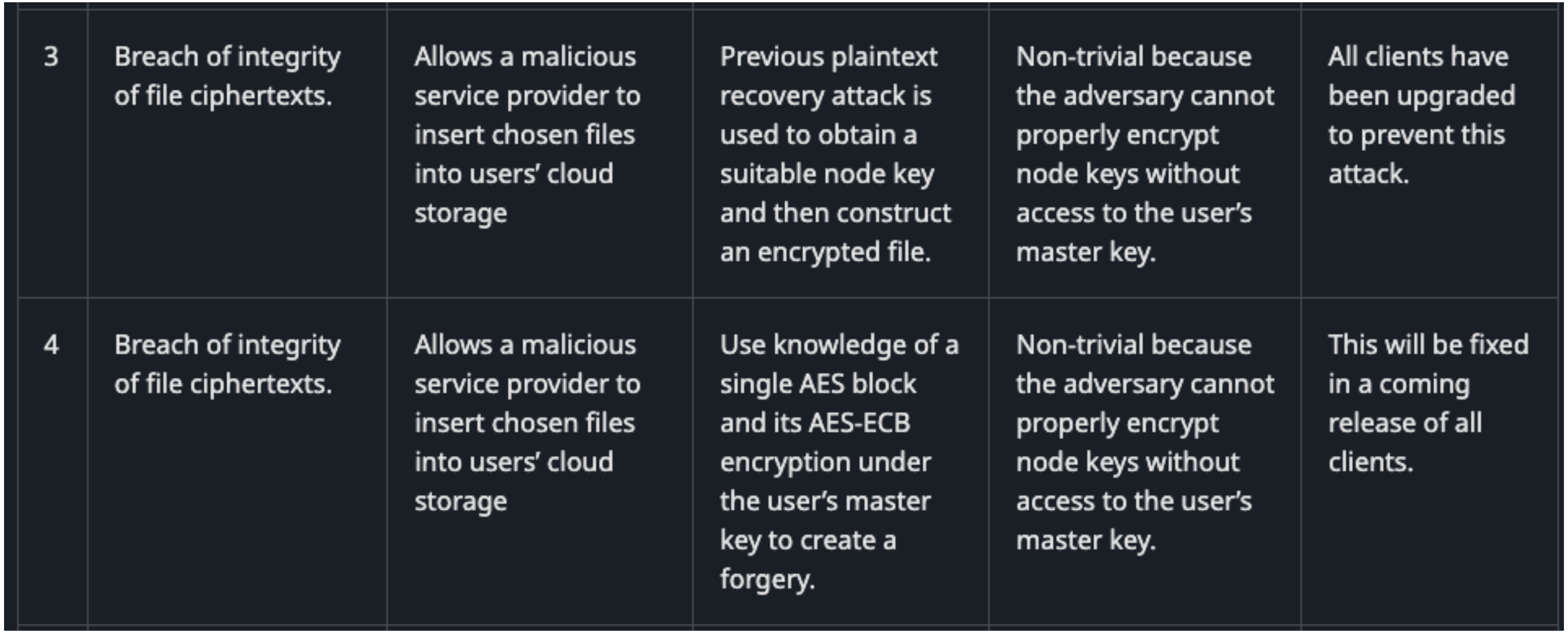
Five Attacks on MEGA



#	Cause	Potential Result	Circumstances	Characterisation	MEGA Response
1	Lack of integrity protection of ciphertexts containing keys.	User-encrypted data could be decrypted.	User has to log in, with their secret password, more than six times	Novel attack vector with known lattice techniques.	All clients have been upgraded to prevent this attack.
2	Lack of integrity protection of ciphertexts containing keys.	User-encrypted data could be decrypted.	User has to log in, with their secret password, more than six times	Entirely novel kind of attack.	All clients have been upgraded to prevent this attack.

blog.mega.io/mega-security-update

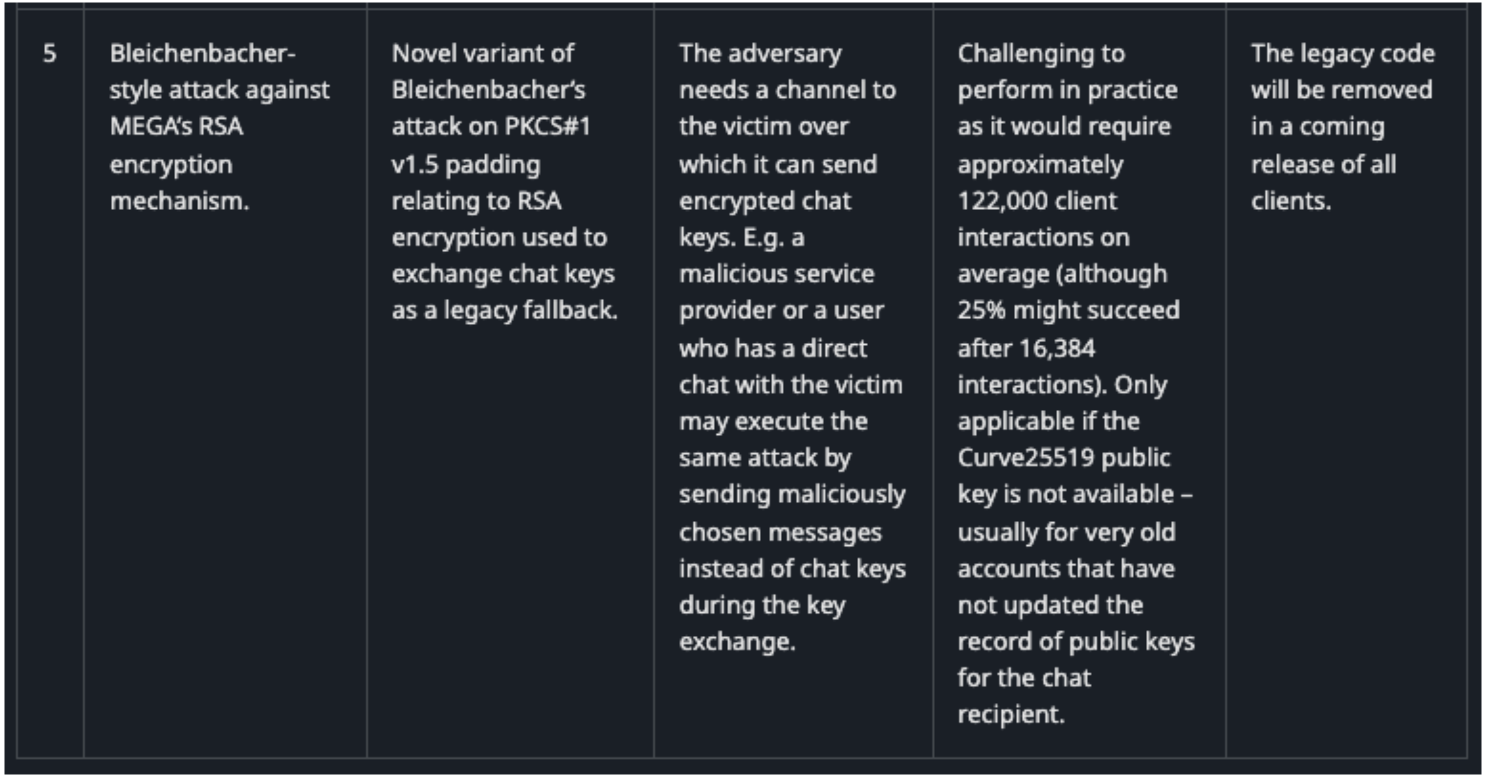
Five Attacks on MEGA



3	Breach of integrity of file ciphertexts.	Allows a malicious service provider to insert chosen files into users' cloud storage	Previous plaintext recovery attack is used to obtain a suitable node key and then construct an encrypted file.	Non-trivial because the adversary cannot properly encrypt node keys without access to the user's master key.	All clients have been upgraded to prevent this attack.
4	Breach of integrity of file ciphertexts.	Allows a malicious service provider to insert chosen files into users' cloud storage	Use knowledge of a single AES block and its AES-ECB encryption under the user's master key to create a forgery.	Non-trivial because the adversary cannot properly encrypt node keys without access to the user's master key.	This will be fixed in a coming release of all clients.

blog.mega.io/mega-security-update

Five Attacks on MEGA



5	Bleichenbacher-style attack against MEGA's RSA encryption mechanism.	Novel variant of Bleichenbacher's attack on PKCS#1 v1.5 padding relating to RSA encryption used to exchange chat keys as a legacy fallback.	The adversary needs a channel to the victim over which it can send encrypted chat keys. E.g. a malicious service provider or a user who has a direct chat with the victim may execute the same attack by sending maliciously chosen messages instead of chat keys during the key exchange.	Challenging to perform in practice as it would require approximately 122,000 client interactions on average (although 25% might succeed after 16,384 interactions). Only applicable if the Curve25519 public key is not available – usually for very old accounts that have not updated the record of public keys for the chat recipient.	The legacy code will be removed in a coming release of all clients.

blog.mega.io/mega-security-update

Attacking the RSA Modulus

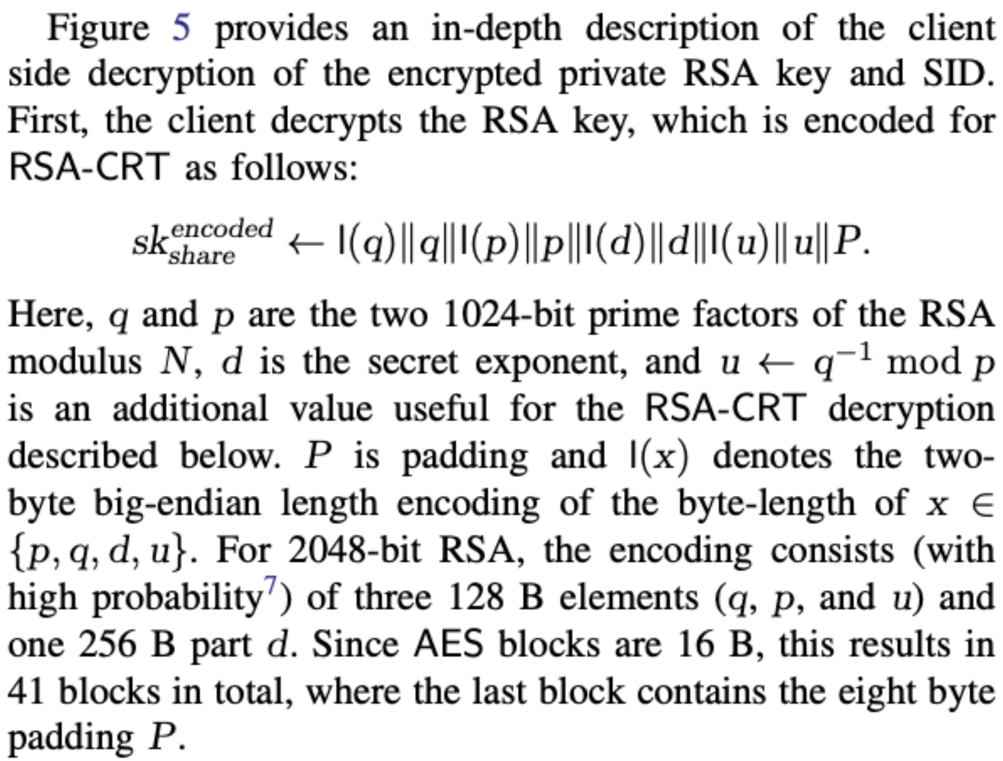
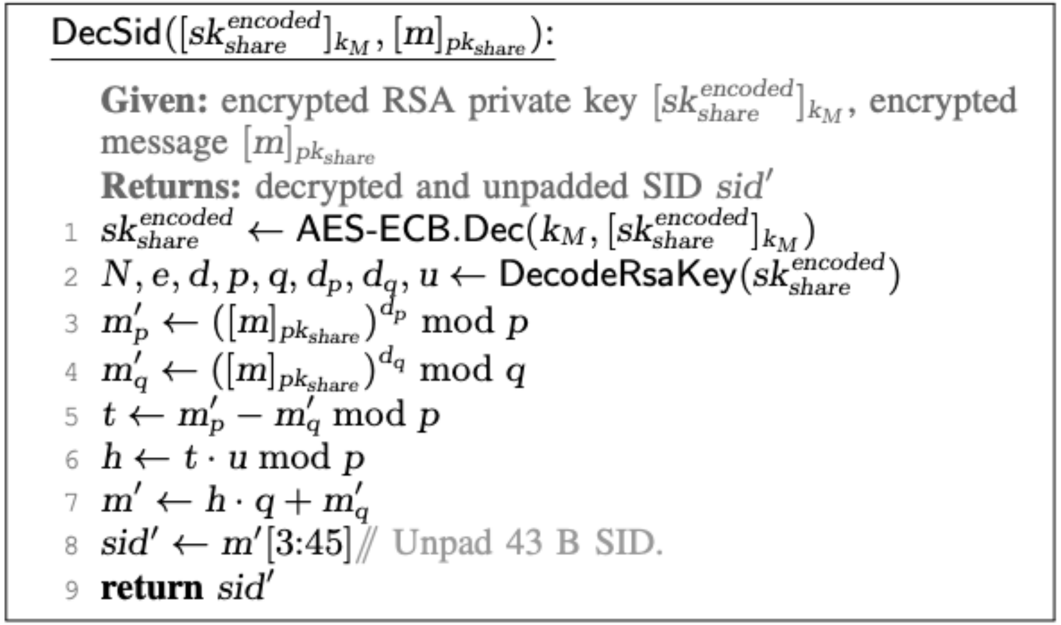


Figure 5 provides an in-depth description of the client side decryption of the encrypted private RSA key and SID. First, the client decrypts the RSA key, which is encoded for RSA-CRT as follows:

$$sk_{share}^{encoded} \leftarrow \mathsf{l}(q)\|q\|\mathsf{l}(p)\|p\|\mathsf{l}(d)\|d\|\mathsf{l}(u)\|u\|P.$$

Here, q and p are the two 1024-bit prime factors of the RSA modulus N, d is the secret exponent, and $u \leftarrow q^{-1} \bmod p$ is an additional value useful for the RSA-CRT decryption described below. P is padding and $\mathsf{l}(x)$ denotes the two-byte big-endian length encoding of the byte-length of $x \in \{p, q, d, u\}$. For 2048-bit RSA, the encoding consists (with high probability[7]) of three 128 B elements (q, p, and u) and one 256 B part d. Since AES blocks are 16 B, this results in 41 blocks in total, where the last block contains the eight byte padding P.

Attacking the RSA Modulus



```
$\mathsf{DecSid}([sk_{share}^{encoded}]_{k_M}, [m]_{pk_{share}})$:
    Given: encrypted RSA private key $[sk_{share}^{encoded}]_{k_M}$, encrypted message $[m]_{pk_{share}}$
    Returns: decrypted and unpadded SID $sid'$
    $sk_{share}^{encoded} \leftarrow \mathsf{AES\text{-}ECB.Dec}(k_M, [sk_{share}^{encoded}]_{k_M})$
    $N, e, d, p, q, d_p, d_q, u \leftarrow \mathsf{DecodeRsaKey}(sk_{share}^{encoded})$
    $m'_p \leftarrow ([m]_{pk_{share}})^{d_p} \bmod p$
    $m'_q \leftarrow ([m]_{pk_{share}})^{d_q} \bmod q$
    $t \leftarrow m'_p - m'_q \bmod p$
    $h \leftarrow t \cdot u \bmod p$
    $m' \leftarrow h \cdot q + m'_q$
    $sid' \leftarrow m'[3{:}45]$ // Unpad 43 B SID.
    return $sid'$
```

Fig. 5. SID decryption during MEGA's client authentication using RSA.

Attacking the RSA Modulus

Overview:

1. Overwrite the key u
2. Distinction Oracle:
   I. If $m < q$, then sid = 0
   II. If $m \geq q$, then sid $\neq 0$
3. Run a binary search starting with bounds $[2^{1023}, 2^{1024} - 1]$.
4. Recover q in 1023 attempts.

Distinction Oracle:

I. Assume $m < q$

Then $m'_p = m$ and $m'_q = m$. Furthermore, we have $t = 0$ and $h = 0$ (independent of u). Then we get $m' = m$ (less than 256^{128}).

Finally, m' is padded with zeroes to 256 bytes, but unpadded by removing the 211 rightmost bytes. Then sid = m[3:45] = 0.

Attacking the RSA Modulus

Overview:

1. Overwrite the key u
2. Distinction Oracle:
   I. If $m < q$, then sid = 0
   II. If $m \geq q$, then sid $\neq 0$
3. Run a binary search starting with bounds $[2^{1023}, 2^{1024} - 1]$.
4. Recover q in 1023 attempts.

Distinction Oracle:

I. Assume $m < q$

II. Assume $m \geq q$

Then $m'_p \neq m$ and $m'_q \neq m$. Furthermore, we have $t \neq 0$ and $h \neq 0$ (w.h.p.) since $u \neq q^{-1} \bmod p$. Then $m' \neq m$ ($\geq 256^{211}$ w.h.p).

Finally, m' is padded with zeroes to 256 bytes, but unpadded by removing the 211 rightmost bytes. Then sid = m[3:45] $\neq 0$.

Attacking the RSA Modulus

- Improvements using a lattice-attack by Gabrielle and Heninger.
- Let $q = q_2 + q_1$ where $q_2 > 2^l$.
- Define polynomials f_1, f_2, f_3.
- Define a 2^l-scaled basis B.
- Run LLL, find $||w||_2 < \det(B)^{1/3}$.
- Ensure $||w||_2 < q$ by $l < \log(N^{1/6})$.
- Corresponding poly $g(q_1) = 0$.
- For RSA-2048 we set $l = 341$.
- Can be further reduced to 512.

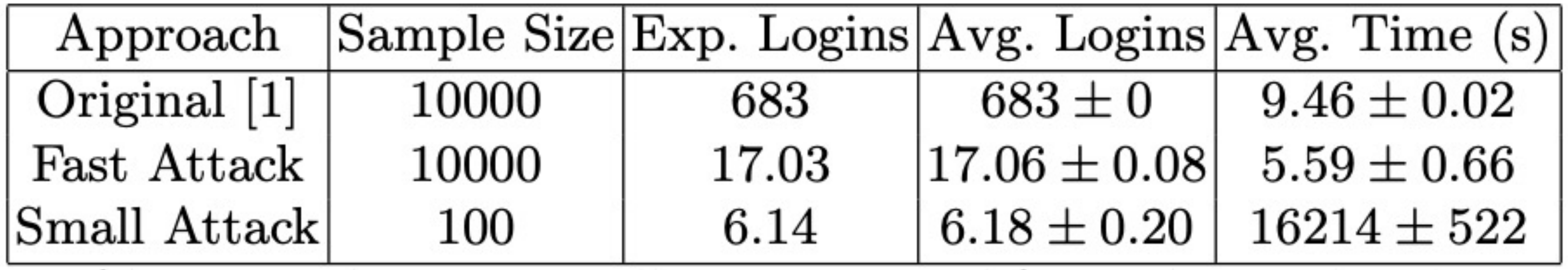
$$\begin{aligned} f_1(x) &= x \cdot (q_2 + x), & f_1(q_1) &= q_1 \cdot (q_2 + q_1) \equiv_q 0 \\ f_2(x) &= q_2 + x, & f_2(q_1) &= q_2 + q_1 \equiv_q 0 \\ f_3(x) &= N, & f_3(q_1) &= N \equiv_q 0 \end{aligned}$$

$L = 2^l$:

$$B = \begin{bmatrix} L^2 & Lq_2 & 0 \\ 0 & L & q_2 \\ 0 & 0 & N \end{bmatrix}$$

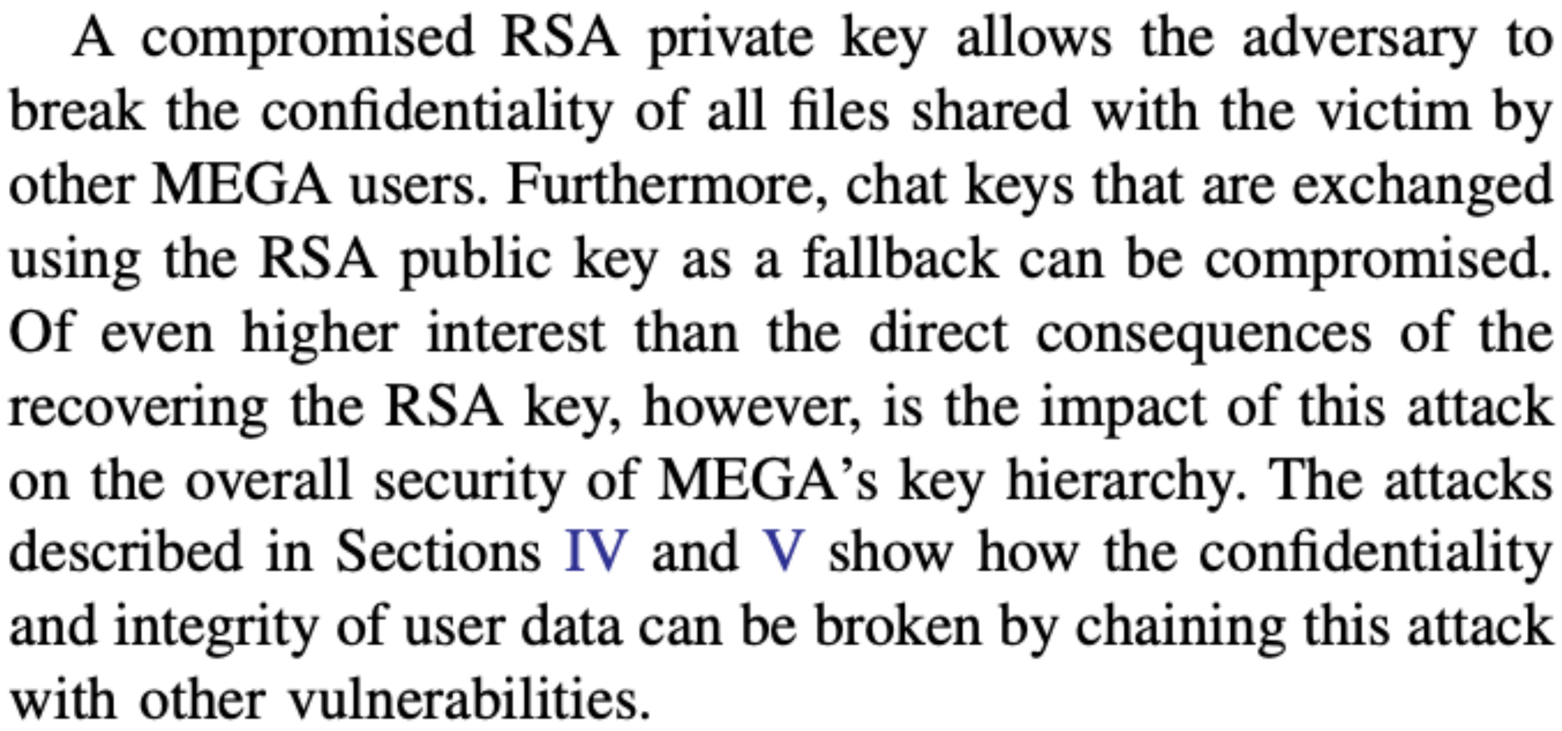
Improved Lattice-Attack

- Recent update by Ryan and Heninger.
- If sid ≠ 0, we actually learn 43 bytes.
- They exploit this to improve the attack:
  - Fast: using approximations of the most significant bytes
  - Small: by brute forcing unknown most significant bytes



Approach	Sample Size	Exp. Logins	Avg. Logins	Avg. Time (s)
Original [1]	10000	683	683 ± 0	9.46 ± 0.02
Fast Attack	10000	17.03	17.06 ± 0.08	5.59 ± 0.66
Small Attack	100	6.14	6.18 ± 0.20	16214 ± 522

Attacking the RSA Modulus



A compromised RSA private key allows the adversary to break the confidentiality of all files shared with the victim by other MEGA users. Furthermore, chat keys that are exchanged using the RSA public key as a fallback can be compromised. Of even higher interest than the direct consequences of the recovering the RSA key, however, is the impact of this attack on the overall security of MEGA's key hierarchy. The attacks described in Sections IV and V show how the confidentiality and integrity of user data can be broken by chaining this attack with other vulnerabilities.

Consequences & Mitigations

- **Plaintext recovery attack**: Building on the previous vulnerability, the malicious service provider can recover any plaintext encrypted with AES-ECB under a user's master key, two blocks at the time, by encrypting q*u.

- **Two integrity attacks**: 1) the plaintext recovery attack allows to obtain a suitable node key and construct encrypted files, 2) exploits a fundamental problem with the method used by MEGA to "obfuscate" file and folder keys before encryption. It needs only knowledge of a single AES block and its AES-ECB encryption under the user's master key to create a forgery.

- **RSA Decryption Attack**: Custom padding for RSA opens for a new Bleichenbacher-type attack using 2^{17} client interactions. Weaker model.

Consequences & Mitigations

- Immediate Countermeasures (backwards compatibility):
  - Adding integrity checks (e.g. HMAC) and individual integrity-keys
  - This efficiently prevents all but the Bleichenbacher-attacks above
  - Key-separation: use a key-derivation key and KDF to generate keys
    - Prevents the plaintext recovery attack when attackers knows RSA key
    - Requires all users to change passwords to rotate their old master keys
  - Use a stricter RSA padding format to increase the attack costs further (~2^{33})

Consequences & Mitigations

- Minimal Countermeasures (more robust, avoids re-encryption):
  - AES-GCM for key ciphertexts to get standardized authenticated encryption
  - RSA-OAEP and separate RSA keys: use standardized RSA and key-separation

- Recommended Countermeasures (complete redesign):
  - File key encryption: Use AES-GCM everywhere, not just for key ciphertexts
    - Use a KDF to derive keys instead of using fixed node keys in file structure
    - Remove “key obfuscation” and start with key rotation for encryption keys
  - Augmented PAKE for authentication: avoid dictionary attacks by third parties

Summary & Conclusions

- Don’t roll your own crypto
- Use standardized cryptography
- Ensure good key hygiene
- Always include integrity checks

THANK YOU!