



MEGA: Malleable Encryption Goes Awry

Modul D3.2

Referent: Dr. Jörg Cosfeld

Kryptographie

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

Kryptographie

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

Bedeutung von Malleable?

Nicht erkennbare Manipulation von Daten. Die Encryption bleibt dabei intakt.

Kryptographie

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

Bedeutung von Awry?

Beschreibt etwas das nicht richtig funktioniert oder Fehler aufweist.

Kryptographie

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

Inhalt dieser Vorlesung:

Diskussion des 20 Seiten Papers und Zusammenfassung des Inhaltes.

Schließung des Kontextes zu besprochenen Themen aus Vorlesung 10.

Kryptographie

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

[Homepage](#) > [People](#) > [Person detail](#)

Prof. Dr. Kenneth Paterson



Address

ETH Zürich
Dep. of Computer Science
Prof. Dr. Kenneth Paterson
Institut f. Informationssicherheit
CNB E 104.2
Universitätstrasse 6
8092 Zürich
Switzerland

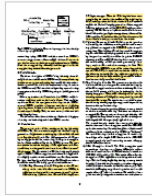
☎ +41 44 632 32 52
✉ kenny.paterson@inf.ethz.ch
📇 V-Card (vcf, 1kb)

- P. Grubbs, M.-S. Lacharité, B. Minaud and K.G. Paterson. Learning to Reconstruct: Statistical Learning Theory and Encrypted Database Attacks. IEEE Symposium on Security and Privacy 2019: 1067-1083.
- P. Leu, M. Singh, M. Roeschlin, K.G. Paterson and S. Capkun. Message Time of Arrival Codes: A Fundamental Primitive for Secure Distance Measurement. IEEE Symposium on Security and Privacy 2020, 500-516.
- J. Massimo and K.G. Paterson. A Performant, Misuse-Resistant API for Primality Testing. ACM CCS 2020: 195-210.
- F. Tramèr, D. Boneh and K.G. Paterson. Remote Side-Channel Attacks on Anonymous Transactions. USENIX Security Symposium 2020: 2739-2756.
- Jean Paul Degabriele, Jérôme Govinden, Felix Günther, Kenneth G. Paterson. The Security of ChaCha20-Poly1305 in the Multi-User Setting. CCS 2021: 1981-2003.
- Lara Bruseghini, Daniel Huigens, Kenneth G. Paterson. Victory by KO: Attacking OpenPGP Using Key Overwriting. ACM CCS 2022: 411-423.
- Martin R. Albrecht, Raphael Eikenberg and Kenneth G. Paterson. Breaking Bridgefy, again: Adopting libsignal is not enough. USENIX Security 22, 249-286.
- Martin R. Albrecht, Lenka Mareková, Kenneth G. Paterson, Igor Stepanovs: Four Attacks and a Proof for Telegram. IEEE Symposium on Security and Privacy 2022: 87-106. Distinguished paper award.
- Matilda Backendal, Felix Günther and Kenneth G. Paterson. Puncturable Key Wrapping and Its Applications. Asiacrypt 2022.
- Mla Filić, Kenneth G. Paterson, Anupama Unnikrishnan and Fernando Virdia. Adversarial Correctness and Privacy for Probabilistic Data Structures. ACM CCS 2022: 1037-1050.
- Paul Grubbs, Varun Maram, Kenneth G. Paterson. Anonymous, Robust Post-quantum Public Key Encryption. EUROCRYPT (3) 2022: 402-432.
- Matilda Backendal, Miro Haller and Kenneth G. Paterson. MEGA: Malleable Encryption Goes Awry. IEEE S&P 2023, to appear.
- Zichen Gui, Kenneth G. Paterson and Sikhar Patranabis. Rethinking Searchable Symmetric Encryption. IEEE S&P 2023, to appear.

Kryptographie



1



2



3



4



5



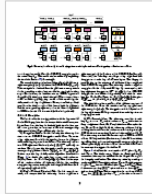
6



7



8



9



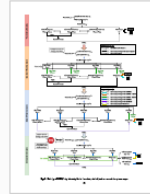
10



11



12



13



14



15



16



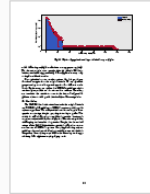
17



18



19



20

Kryptographie



Give me a summary of the paper MEGA: Malleable Encryption Goes Awry



Kryptographie



Give me a summary of the paper MEGA: Malleable Encryption Goes Awry



MEGA: Malleable Encryption Goes Awry is a paper that discusses the security of MEGA, a cloud storage and collaboration platform that claims to offer user-controlled, end-to-end encryption (UCE). The paper discusses the security of MEGA under the assumption that the service provider itself could potentially be adversarial, and that MEGA's systems could be compromised by malicious third parties. The paper presents five distinct attacks against MEGA that allow for the compromise of the confidentiality of user files and the insertion of malicious files by an attacker. Four of these attacks are practical and have been responsibly disclosed to MEGA, which is working on remediation. The paper also discusses the challenges of deploying cryptography at a large scale under strong threat models and provides recommendations for countermeasures. The paper concludes that there are significant shortcomings in MEGA's cryptographic architecture, and that further work is needed to improve the security of the platform.



Kryptographie



Give me a summary of the paper MEGA: Malleable Encryption Goes Awry



MEGA: Malleable Encryption Goes Awry is a paper that discusses the security of MEGA, a cloud storage and collaboration platform that claims to offer user-controlled, end-to-end encryption (UCE). The paper discusses the security of MEGA under the assumption that the service provider itself could potentially be adversarial, and that MEGA's systems could be compromised by malicious third parties. The paper presents five distinct attacks against MEGA that allow for the compromise of the confidentiality of user files and the insertion of malicious files by an attacker. Four of these attacks are practical and have been responsibly disclosed to MEGA, which is working on remediation. The paper also discusses the challenges of deploying cryptography at a large scale under strong threat models and provides recommendations for countermeasures. The paper concludes that there are significant shortcomings in MEGA's cryptographic architecture, and that further work is needed to improve the security of the platform.



Kryptographie



Give me a summary of the paper MEGA: Malleable Encryption Goes Awry



MEGA: Malleable Encryption Goes Awry is a paper that discusses the security of MEGA, a cloud storage and collaboration platform that claims to offer user-controlled, end-to-end encryption (UCE). The paper discusses the security of MEGA under the assumption that the service provider itself could potentially be adversarial, and that MEGA's systems could be compromised by malicious third parties. The paper presents five distinct attacks against MEGA that allow for the compromise of the confidentiality of user files and the insertion of malicious files by an attacker. Four of these attacks are practical and have been responsibly disclosed to MEGA, which is working on remediation. The paper also discusses the challenges of deploying cryptography at a large scale under strong threat models and provides recommendations for countermeasures. The paper concludes that there are significant shortcomings in MEGA's cryptographic architecture, and that further work is needed to improve the security of the platform.



Kryptographie



Give me a summary of the paper MEGA: Malleable Encryption Goes Awry



MEGA: Malleable Encryption Goes Awry is a paper that discusses the security of MEGA, a cloud storage and collaboration platform that claims to offer user-controlled, end-to-end encryption (UCE). The paper discusses the security of MEGA under the assumption that the service provider itself could potentially be adversarial, and that MEGA's systems could be compromised by malicious third parties. The paper presents five distinct attacks against MEGA that allow for the compromise of the confidentiality of user files and the insertion of malicious files by an attacker. Four of these attacks are practical and have been responsibly disclosed to MEGA, which is working on remediation. The paper also discusses the challenges of deploying cryptography at a large scale under strong threat models and provides recommendations for countermeasures. The paper concludes that there are significant shortcomings in MEGA's cryptographic architecture, and that further work is needed to improve the security of the platform.



Kryptographie



Give me a summary of the paper MEGA: Malleable Encryption Goes Awry



MEGA: Malleable Encryption Goes Awry is a paper that discusses the security of MEGA, a cloud storage and collaboration platform that claims to offer user-controlled, end-to-end encryption (UCE). The paper discusses the security of MEGA under the assumption that the service provider itself could potentially be adversarial, and that MEGA's systems could be compromised by malicious third parties. The paper presents five distinct attacks against MEGA that allow for the compromise of the confidentiality of user files and the insertion of malicious files by an attacker. Four of these attacks are practical and have been responsibly disclosed to MEGA, which is working on remediation. The paper also discusses the challenges of deploying cryptography at a large scale under strong threat models and provides recommendations for countermeasures. The paper concludes that there are significant shortcomings in MEGA's cryptographic architecture and that further work is needed to improve the security of the platform.



Kryptographie

Was erfahren wir über das
Paper wenn wir den Abstract
lesen?

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

Abstract—MEGA is a leading cloud storage platform with more than 250 million users and 1000 Petabytes of stored data. MEGA claims to offer user-controlled, end-to-end security. This is achieved by having all data encryption and decryption operations done on MEGA clients, under the control of keys that are only available to those clients. This is intended to protect MEGA users from attacks by MEGA itself, or by adversaries who have taken control of MEGA's infrastructure.

We provide a detailed analysis of MEGA's use of cryptography in such a malicious server setting. We present five distinct attacks against MEGA, which together allow for a full compromise of the confidentiality of user files. Additionally, the integrity of user data is damaged to the extent that an attacker can insert malicious files of their choice which pass all authenticity checks of the client. We built proof-of-concept versions of all the attacks. Four of the five attacks are eminently practical. They have all been responsibly disclosed to MEGA and remediation is underway.

Taken together, our attacks highlight significant shortcomings in MEGA's cryptographic architecture. We present immediately deployable countermeasures, as well as longer-term recommendations. We also provide a broader discussion of the challenges of cryptographic deployment at massive scale under strong threat models.

Kryptographie

Was erfahren wir über das Paper wenn wir den Abstract lesen?

- Es gibt wenig Unterschiede zu der Zusammenfassung der KI.

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@alumni1.ethz.ch

Abstract—MEGA is a leading cloud storage platform with more than 250 million users and 1000 Petabytes of stored data. MEGA claims to offer user-controlled, end-to-end security. This is achieved by having all data encryption and decryption operations done on MEGA clients, under the control of keys that are only available to those clients. This is intended to protect MEGA users from attacks by MEGA itself, or by adversaries who have taken control of MEGA's infrastructure.

We provide a detailed analysis of MEGA's use of cryptography in such a malicious server setting. We present five distinct attacks against MEGA, which together allow for a full compromise of the confidentiality of user files. Additionally, the integrity of user data is damaged to the extent that an attacker can insert malicious files of their choice which pass all authenticity checks of the client. We built proof-of-concept versions of all the attacks. Four of the five attacks are eminently practical. They have all been responsibly disclosed to MEGA and remediation is underway.




Taken together, our attacks highlight significant shortcomings in MEGA's cryptographic architecture. We present immediately deployable countermeasures, as well as longer-term recommendations. We also provide a broader discussion of the challenges of cryptographic deployment at massive scale under strong threat models.

Kryptographie

Was erfahren wir über das Paper wenn wir den Abstract lesen?

- Es gibt wenig Unterschiede zu der Zusammenfassung der KI.

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



Wie viele Kapitel umfasst das Paper?



I'm sorry, but I am not able to understand your question. Could you please provide more context or clarify your question in English?



Wie viele Seiten umfasst das Paper, MEGA: Malleable Encryption Goes Awry?





I'm sorry, but I don't have information about the length of the paper MEGA: Malleable Encryption Goes Awry. Could you please provide more context or clarify your question?



Kryptographie



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

Abstract—MEGA is a leading cloud storage platform with more than 250 million users and 1000 Petabytes of stored data. MEGA claims to offer user-controlled, end-to-end security. This is achieved by having all data encryption and decryption operations done on MEGA clients, under the control of keys that are only available to those clients. This is intended to protect MEGA users from attacks by MEGA itself, or by adversaries who have taken control of MEGA's infrastructure.

We provide a detailed analysis of MEGA's use of cryptography in such a malicious server setting. We present five distinct attacks against MEGA, which together allow for a full compromise of the confidentiality of user files. Additionally, the integrity of user data is damaged to the extent that an attacker can insert malicious files of their choice which pass all authenticity checks of the client. We built proof-of-concept versions of all the attacks. Four of the five attacks are eminently practical. They have all been responsibly disclosed to MEGA and remediation is underway.

Taken together, our attacks highlight significant shortcomings in MEGA's cryptographic architecture. We present immediately deployable countermeasures, as well as longer-term recommendations. We also provide a broader discussion of the challenges of cryptographic deployment at massive scale under strong threat models.

1. INTRODUCTION

The cloud – for outsourcing of both computation and data storage – has become a very popular approach to address scaling and management problems in IT. This applies to both enterprise and consumer domains. In the latter case, the market offers a myriad of different cloud services, with products having different combinations of storage, computation and collaboration features, and making a range of security and privacy claims. The consumer storage market alone was valued at USD 13.6 billion in 2021.

As a prominent example, MEGA¹ is a cloud storage and collaboration platform founded in 2013 offering “secure storage and communication” services. With over 250 million registered users, 10 million daily active users [1] and 1000 PB of stored data [2], MEGA is a significant player in the consumer domain. What sets them apart from their competitors such as Dropbox, Google Drive, iCloud and Microsoft OneDrive is the claimed security guarantees: MEGA advertise themselves as “the privacy company” and promise user-controlled end-to-end encryption (UCE).

UCE refers to the fact that data uploaded to the MEGA cloud is encrypted, and that only the user who owns the data has access to the key (derived from the user's password) needed to decrypt. Thus, MEGA's main selling point is

confidentiality of user data even against MEGA themselves, as showcased in the following quote from their website [3]:

“MEGA does not have access to your password or your data. Using a strong and unique password will ensure that your data is protected from being hacked and gives you total confidence that your information will remain just that – yours.”

This implies a threat model in which the service provider itself should be considered potentially adversarial, and yet the service should remain secure. All the service is then trusted for its availability. This adversarial model provides an interesting setting for cryptanalysis: not only does the adversary have access to encrypted user keys and data, it can also interact with users through legitimate channels during steps like user authentication and file access.

This may seem a very strong adversarial model. However, we stress that it is consistent with the security claims made by MEGA themselves. Moreover, we must consider the possibility that even if MEGA is not adversarial, their systems may have been compromised by malicious third parties, for example nation state security agencies or hacking groups, who wish to gain access to users' data and files. Indeed, the sheer size of MEGA – and the likelihood of it attracting users who wish to protect highly sensitive data precisely because of the security the service claims to offer – surely make MEGA an attractive target. Additionally, UCE should ensure that MEGA cannot be coerced into revealing user data, e.g. through subpoenas, since they are technically unable to do so.

In this work, we review the security of MEGA in this threat model and find significant issues in how it uses cryptography. These lead to devastating attacks on the confidentiality and integrity of user data in the MEGA cloud.


A. The MEGA Key Hierarchy

MEGA's approach to UCE begins with the user password, PW, which acts as the root of the key hierarchy depicted in Figure 1. The MEGA client derives an authentication key and an encryption key from the password. The authentication key is used to identify users to MEGA. The encryption key encrypts a randomly generated master key, which in turn encrypts other key material of the user. Every account has a set of asymmetric keys: an RSA key pair for sharing data, a Curve25519 key pair for exchanging chat keys for MEGA's chat functionality, and an Ed25519 key pair for signing the other keys. Furthermore, the client generates a new key for every file or folder (collectively referred to as *nodes*) uploaded by the user. All keys are encrypted by the client with the

¹<https://dataintel.com/report/consumer-cloud-storage-services-market/>
²<https://mega.io>

Kryptographie

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



Abstract—MEGA is a leading cloud storage platform with more than 250 million users and 1000 Petabytes of stored data. MEGA claims to offer user-controlled, end-to-end security. This is achieved by having all data encryption and decryption operations done on MEGA clients, under the control of keys that are only available to those clients. This is intended to protect MEGA users from attacks by MEGA itself, or by adversaries who have taken control of MEGA's infrastructure.

We provide a detailed analysis of MEGA's use of cryptography in such a malicious server setting. We present five distinct attacks against MEGA, which together allow for a full compromise of the confidentiality of user files. Additionally, the integrity of user data is damaged to the extent that an attacker can insert malicious files of their choice which pass all authenticity checks of the client. We built proof-of-concept versions of all the attacks. Four of the five attacks are eminently practical. They have all been responsibly disclosed to MEGA and remediation is underway.

Taken together, our attacks highlight significant shortcomings in MEGA's cryptographic architecture. We present immediately deployable countermeasures, as well as longer-term recommendations. We also provide a broader discussion of the challenges of cryptographic deployment at massive scale under strong threat models.

confidentiality of user data even against MEGA themselves, as showcased in the following quote from their website [3]:

"MEGA does not have access to your password or your data. Using a strong and unique password will ensure that your data is protected from being hacked and gives you total confidence that your information will remain just that – yours."

This implies a threat model in which the service provider itself should be considered potentially adversarial, and yet the service should remain secure. All the service is then trusted for its availability. This adversarial model provides an interesting setting for cryptanalysis: not only does the adversary have access to encrypted user keys and data, it can also interact with users through legitimate channels during steps like user authentication and file access.

This may seem a very strong adversarial model. However, we stress that it is consistent with the security claims made by MEGA themselves. Moreover, we must consider the possibility that even if MEGA is not adversarial, their systems



The paper MEGA: Malleable Encryption Goes Awry discusses the security of MEGA, a cloud storage and collaboration platform that claims to offer user-controlled, end-to-end encryption (UCE). UCE refers to the fact that data uploaded to the MEGA cloud is encrypted, and that only the user who owns the data has access to the key needed to decrypt it. This is intended to protect MEGA users from attacks by MEGA itself or by adversaries who have taken control of MEGA's infrastructure. The paper provides a detailed analysis of MEGA's use of cryptography in a malicious server setting and identifies significant issues in the platform's cryptographic architecture. These issues lead to devastating attacks on the confidentiality and integrity of user data in the MEGA cloud. The paper also discusses the challenges of deploying cryptography at a large scale under strong threat models and provides recommendations for countermeasures.

<https://dataintel.com/report/consumer-cloud-storage-services-market/>
<https://mega.io>

every file or folder (collectively referred to as *nodes*) uploaded by the user. All keys are encrypted by the client with the

Write a detailed summary with emphasis on encryption of this text:

I. INTRODUCTION

The cloud – for outsourcing of both computation and data storage – has become a very popular approach to address scaling and management problems in IT. This applies to both enterprise and consumer domains. In the latter case, the market offers a myriad of different cloud services, with products




...

Kryptographie

- Gute Zusammenfassung aber wichtige Punkte sind nicht aufgeführt.



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

Abstract—MEGA is a leading cloud storage platform with more than 250 million users and 1000 Petabytes of stored data. MEGA claims to offer user-controlled, end-to-end security. This is achieved by having all data encryption and decryption operations done on MEGA clients, under the control of keys that are only available to those clients. This is intended to protect MEGA users from attacks by MEGA itself, or by adversaries who have taken control of MEGA's infrastructure.

We provide a detailed analysis of MEGA's use of cryptography in such a malicious server setting. We present five distinct attacks against MEGA, which together allow for a full compromise of the confidentiality of user files. Additionally, the integrity of user data is damaged to the extent that an attacker can insert malicious files of their choice which pass all authenticity checks of the client. We built proof-of-concept versions of all the attacks. Four of the five attacks are eminently practical. They have all been responsibly disclosed to MEGA and remediation is underway.

Taken together, our attacks highlight significant shortcomings in MEGA's cryptographic architecture. We present immediately deployable countermeasures, as well as longer-term recommendations. We also provide a broader discussion of the challenges of cryptographic deployment at massive scale under strong threat models.

confidentiality of user data even against MEGA themselves, as showcased in the following quote from their website [3]:

"MEGA does not have access to your password or your data. Using a strong and unique password will ensure that your data is protected from being hacked and gives you total confidence that your information will remain just that – yours."

This implies a threat model in which the service provider itself should be considered potentially adversarial, and yet the service should remain secure. All the service is then trusted for its availability. This adversarial model provides an interesting setting for cryptanalysis: not only does the adversary have access to encrypted user keys and data, it can also interact with users through legitimate channels during steps like user authentication and file access.

This may seem a very strong adversarial model. However, we stress that it is consistent with the security claims made by MEGA themselves. Moreover, we must consider the possibility that even if MEGA is not adversarial, their systems



The paper MEGA: Malleable Encryption Goes Awry discusses the security of MEGA, a cloud storage and collaboration platform that claims to offer user-controlled, end-to-end encryption (UCE). UCE refers to the fact that data uploaded to the MEGA cloud is encrypted, and that only the user who owns the data has access to the key needed to decrypt it. This is intended to protect MEGA users from attacks by MEGA itself or by adversaries who have taken control of MEGA's infrastructure. The paper provides a detailed analysis of MEGA's use of cryptography in a malicious server setting and identifies significant issues in the platform's cryptographic architecture. These issues lead to devastating attacks on the confidentiality and integrity of user data in the MEGA cloud. The paper also discusses the challenges of deploying cryptography at a large scale under strong threat models and provides recommendations for countermeasures.

<https://dataintel.com/report/consumer-cloud-storage-services-market/>
<https://mega.io>

every file or folder (collectively referred to as *nodes*) uploaded by the user. All keys are encrypted by the client with the

Write a detailed summary with emphasis on encryption of this text:

I. INTRODUCTION

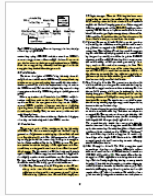
The cloud – for outsourcing of both computation and data storage – has become a very popular approach to address scaling and management problems in IT. This applies to both enterprise and consumer domains. In the latter case, the market offers a myriad of different cloud services, with products

...

Kryptographie



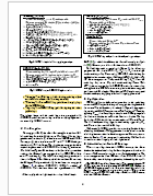
1



2



3



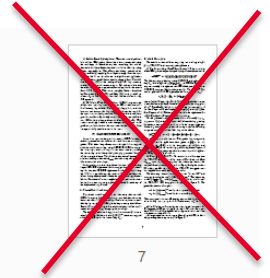
4



5



6



7



8



9



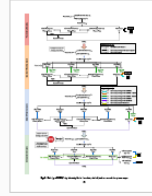
10



11



12



13



14



15



16



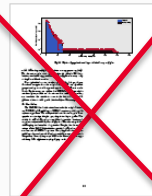
17



18

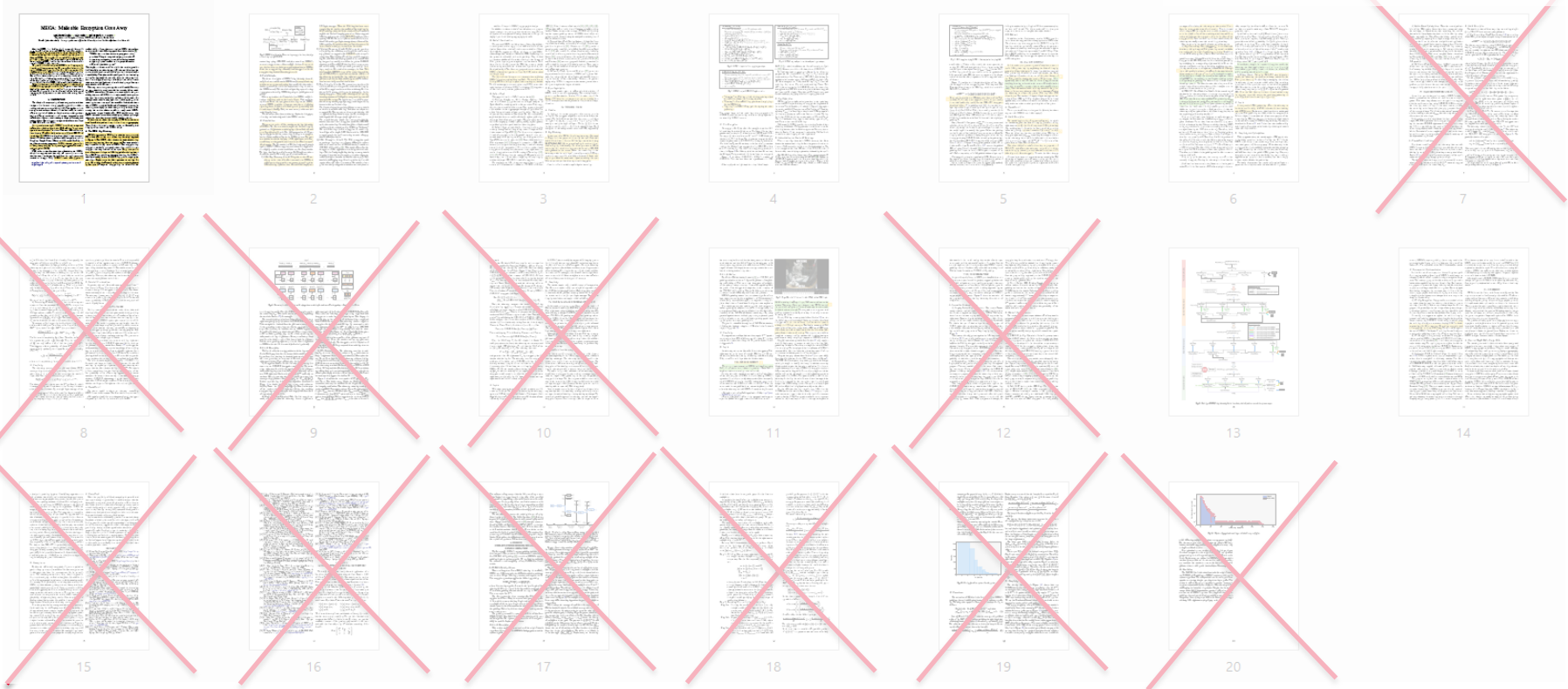


19

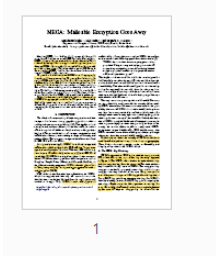


20

Kryptographie



Kryptographie

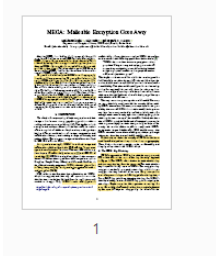


Was nehmen wir als Infos von Seite eins mit?

- Mega ist eine cloud storage Dienst
- 10 Mio. tägliche Nutzer
- 1000 PB an Daten

In this work, we review the security of MEGA in this threat model and find significant issues in how it uses cryptography.

Kryptographie



Was nehmen wir als Infos von Seite eins mit?

- Mega ist eine cloud storage Dienst
- 10 Mio. tägliche Nutzer
- 1000 PB an Daten

MEGA's approach to UCE begins with the user password, PW, which acts as the root of the key hierarchy depicted in Figure 1. The MEGA client derives an authentication key and an encryption key from the password. The authentication

Kryptographie

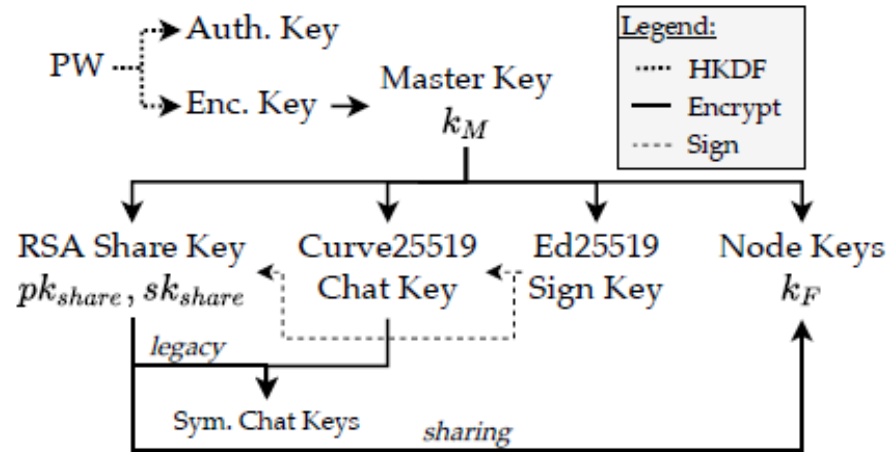
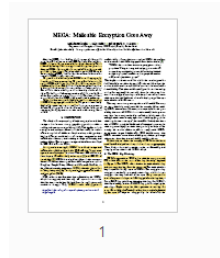
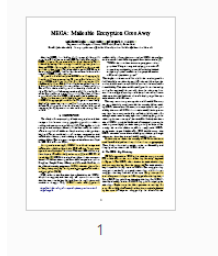


Fig. 1. MEGA's key hierarchy. The master key encrypts the share, chat, sign and node keys using AES-ECB.

MEGA's approach to UCE begins with the user password, PW, which acts as the root of the key hierarchy depicted in Figure 1. The MEGA client derives an authentication key and an encryption key from the password. The authentication

Kryptographie



Was ist die wichtigste Info von Fig. 1?

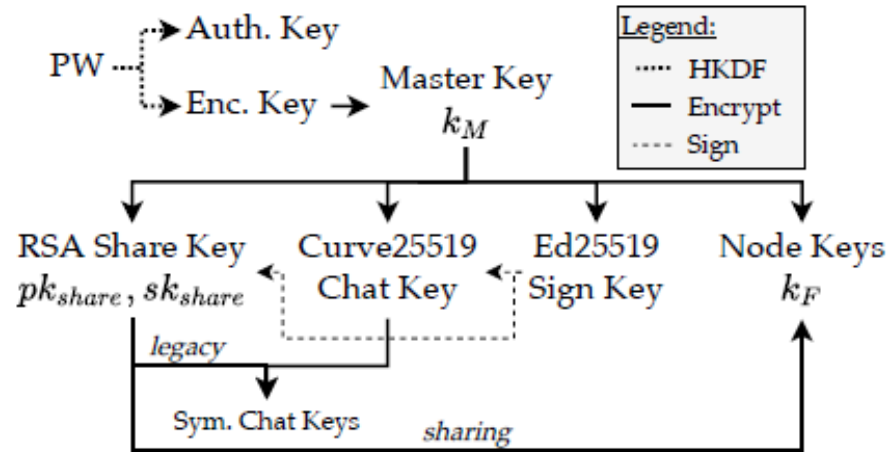
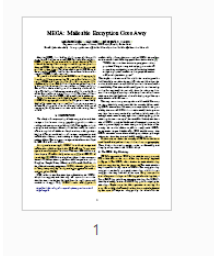


Fig. 1. MEGA's key hierarchy. The master key encrypts the share, chat, sign and node keys using AES-ECB.

MEGA's approach to UCE begins with the user password, PW, which acts as the root of the key hierarchy depicted in Figure 1. The MEGA client derives an authentication key and an encryption key from the password. The authentication

Kryptographie



Was ist die wichtigste Info von Fig. 1?

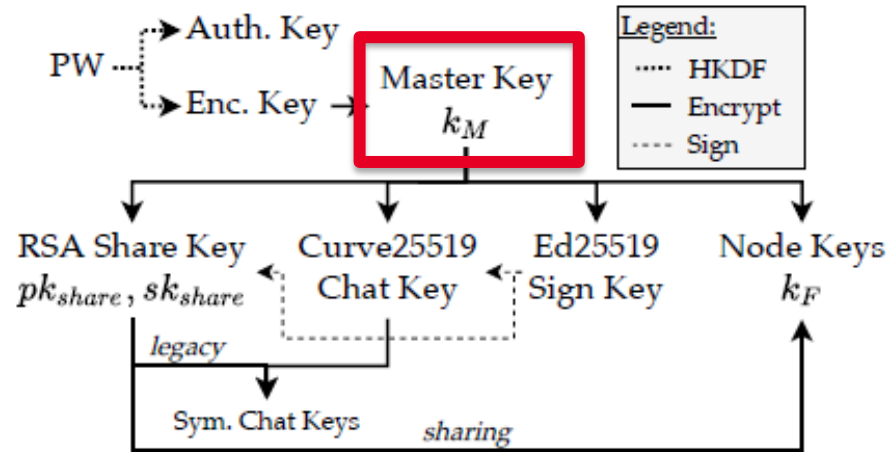


Fig. 1. MEGA's key hierarchy. The master key encrypts the share, chat, sign and node keys using AES-ECB.

MEGA's approach to UCE begins with the user password, PW, which acts as the root of the key hierarchy depicted in Figure 1. The MEGA client derives an authentication key and an encryption key from the password. The authentication

Kryptographie

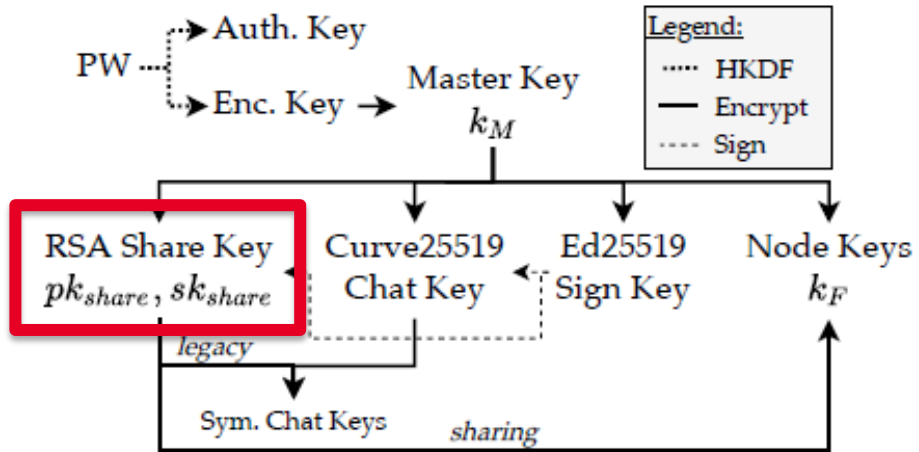
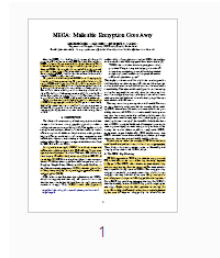
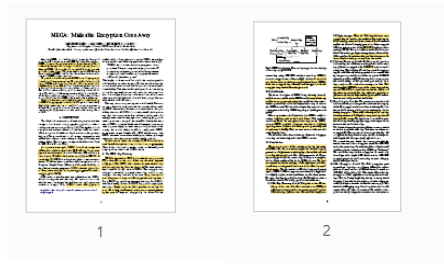


Fig. 1. MEGA's key hierarchy. The master key encrypts the share, chat, sign and node keys using AES-ECB.

encrypts other key material of the user. Every account has a set of asymmetric keys: an RSA key pair for sharing data, a

Kryptographie



Explanation to Curve25519

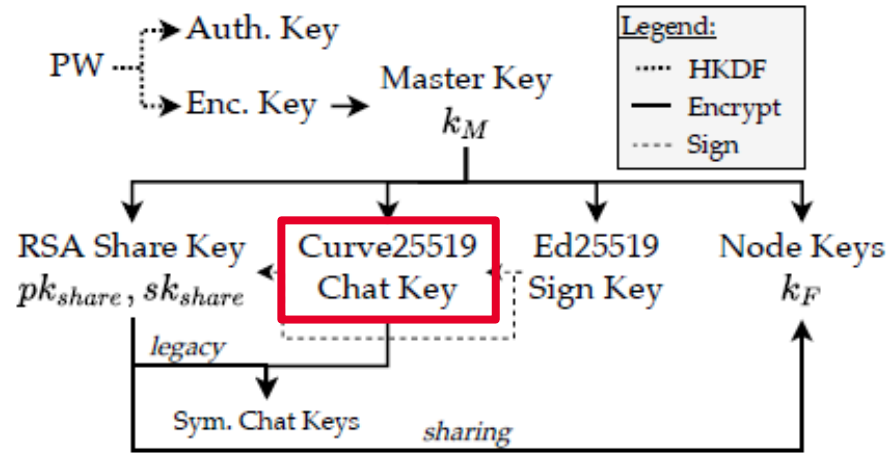


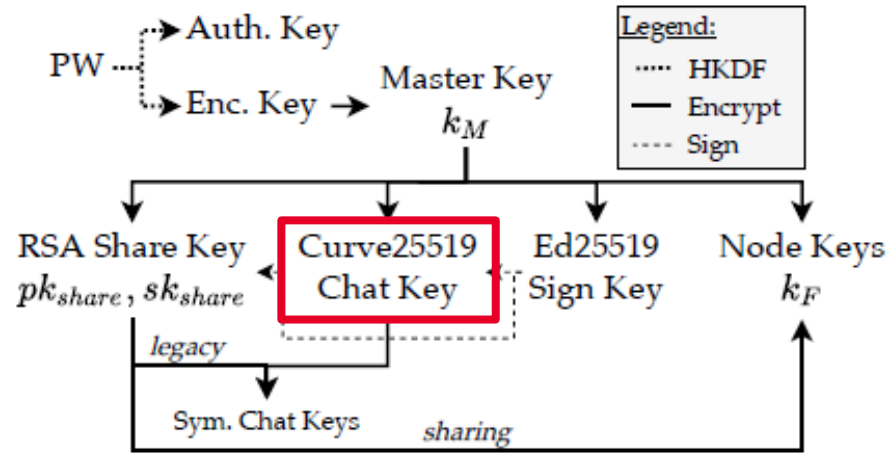
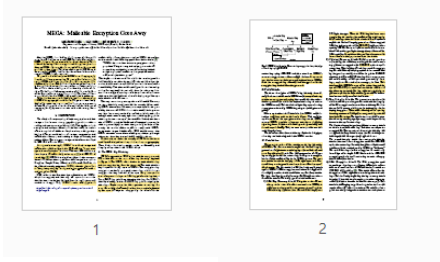
Fig. 1. MEGA's key hierarchy. The master key encrypts the share, chat, sign and node keys using AES-ECB.



Curve25519 is an elliptic curve that can be used for public-key cryptography. It is a type of public-key algorithm that is based on the algebraic structure of an elliptic curve over a finite field.



Kryptographie



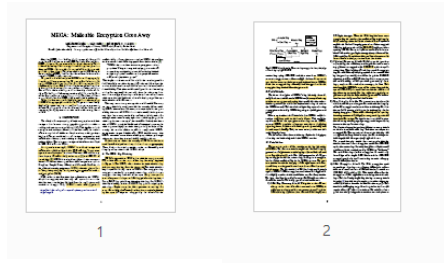
show figures to Curve25519



Fig. 1. MEGA's key hierarchy. The master key encrypts the share, chat, sign and node keys using AES-ECB.

Kann keine Figures oder Plots anzeigen.

Kryptographie



plot(y**2 = x^3 + 486662x^2 + x)

NATURAL LANGUAGE

MATH INPUT

EXTENDED KEYBOARD

EXAMPLES

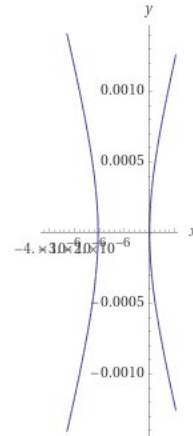
UPLOAD

RANDOM

Input interpretation

plot $y^2 = x^3 + 486662x^2 + x$

Implicit plot



Download Page

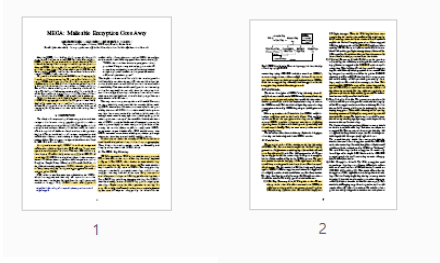
POWERED BY THE WOLFRAM LANGUAGE



show figures to Curve25519



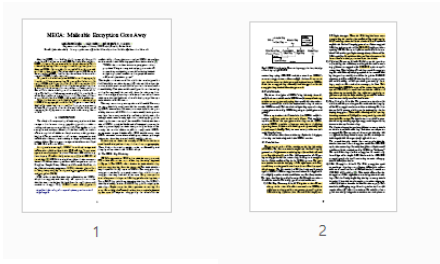
Kryptographie



Was nehmen wir als Infos von Seite zwei mit?

We present a series of five attacks on the key hierarchy of MEGA. The first two attacks exploit the lack of integrity

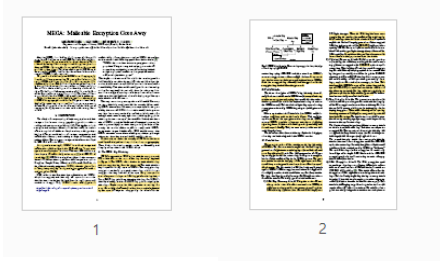
Kryptographie



Was nehmen wir als Infos von Seite zwei mit?

We present a series of five attacks on the key hierarchy of MEGA. The first two attacks exploit the lack of integrity protection of ciphertexts containing keys (henceforth referred to as *key ciphertexts*), and allow full compromise of all user keys encrypted with the master key, leading to a complete

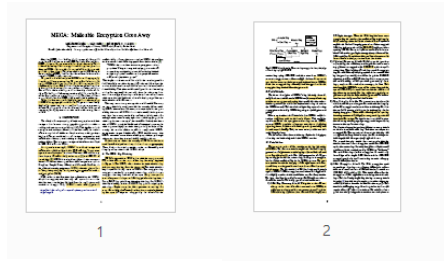
Kryptographie



Was nehmen wir als Infos von Seite zwei mit?

- 1) **RSA Key Recovery Attack.** Using the session ID exchange at the start of a client connection to MEGA, a malicious service provider can recover a user's private RSA share key (used to share file and folder keys) over

Kryptographie



Was nehmen wir als Infos von Seite zwei mit?

- 1) **RSA Key Recovery Attack.** Using the session ID exchange at the start of a client connection to MEGA, a malicious service provider can recover a user's private RSA share key (used to share file and folder keys) over

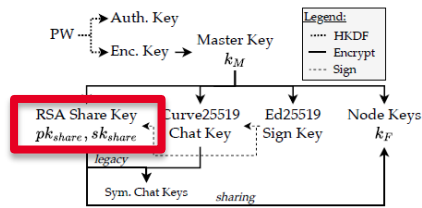
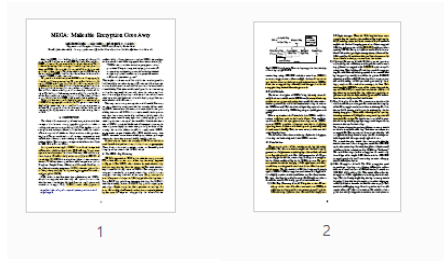


Fig. 1. MEGA's key hierarchy. The master key encrypts the share, chat, sign and node keys using AES-ECB.

512 login attempts. When the RSA key has been compromised by the attacker, the confidentiality and integrity of all node keys shared with the victim is lost. Our attack

Kryptographie



Was nehmen wir als Infos von Seite zwei mit?

- 1) **RSA Key Recovery Attack.** Using the session ID exchange at the start of a client connection to MEGA, a malicious service provider can recover a user's private RSA share key (used to share file and folder keys) over

512 login attempts. When the RSA key has been compromised by the attacker, the confidentiality and integrity of all node keys shared with the victim is lost. Our attack

RSA modulus. It combines this novel attack vector with known lattice techniques to accelerate the attack.

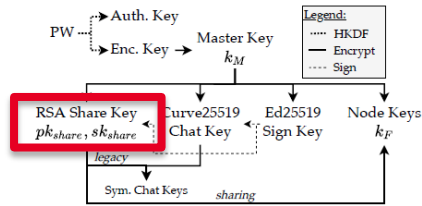
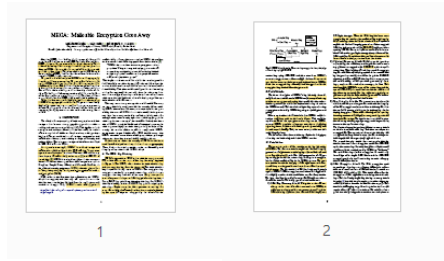


Fig. 1. MEGA's key hierarchy. The master key encrypts the share, chat, sign and node keys using AES-ECB.

Kryptographie



Was nehmen wir als Infos von Seite zwei mit?

- 1) **RSA Key Recovery Attack.** Using the session ID exchange at the start of a client connection to MEGA, a malicious service provider can recover a user's private RSA share key (used to share file and folder keys) over

512 login attempts. When the RSA key has been compromised by the attacker, the confidentiality and integrity of all node keys shared with the victim is lost. Our attack

RSA modulus. It combines this novel attack vector with known lattice techniques to accelerate the attack.

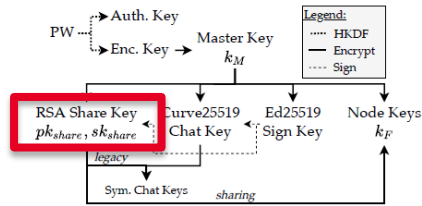
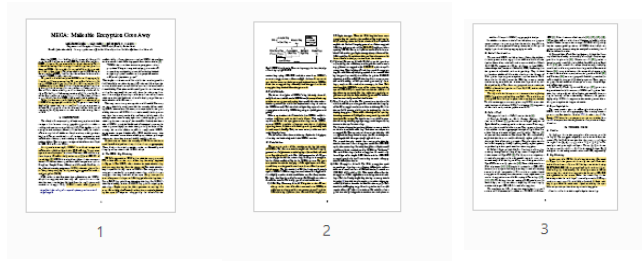


Fig. 1. MEGA's key hierarchy. The master key encrypts the share, chat, sign and node keys using AES-ECB.

Kryptographie



Was nehmen wir als Infos von Seite drei mit?

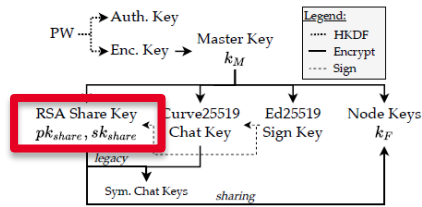
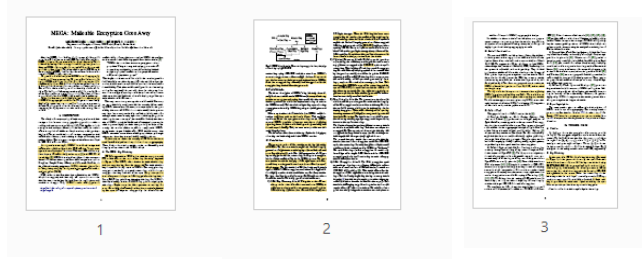


Fig. 1. MEGA's key hierarchy. The master key encrypts the share, chat, sign and node keys using AES-ECB.

fix significantly differs from our proposed countermeasures. MEGA released their patches on June 21, 2022, and awarded us a bug bounty.

We only worked with our own test account when exploring MEGA's services and building our PoCs. We avoided overloading MEGA with login requests when running our attacks. We did not attempt to reverse engineer any MEGA server-side

Kryptographie



Was nehmen wir als Infos von Seite drei mit?

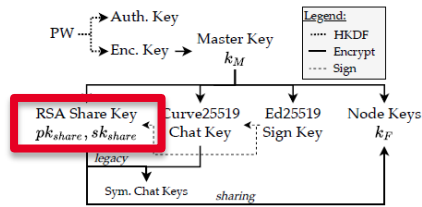


Fig. 1. MEGA's key hierarchy. The master key encrypts the share, chat, sign and node keys using AES-ECB.

fix significantly differs from our proposed countermeasures. MEGA released their patches on June 21, 2022, and awarded us a bug bounty.

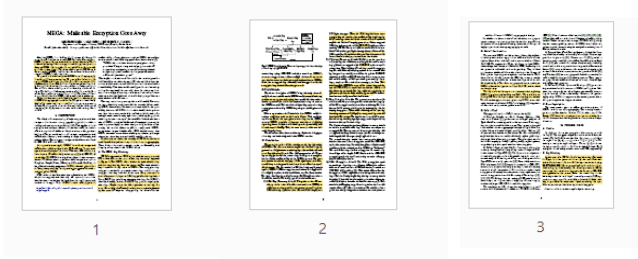
We only worked with our own test account when exploring MEGA's services and building our PoCs. We avoided overloading MEGA with login requests when running our attacks. We did not attempt to reverse engineer any MEGA server-side



Did mega fix or patch the shortcomings



Kryptographie



Was nehmen wir als Infos von Seite drei mit?

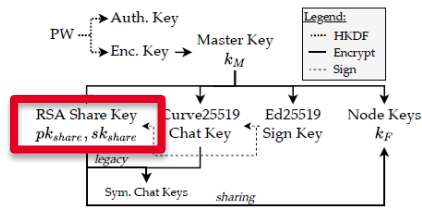


Fig. 1. MEGA's key hierarchy. The master key encrypts the share, chat, sign and node keys using AES-ECB.

fix significantly differs from our proposed countermeasures. MEGA released their patches on June 21, 2022, and awarded us a bug bounty.

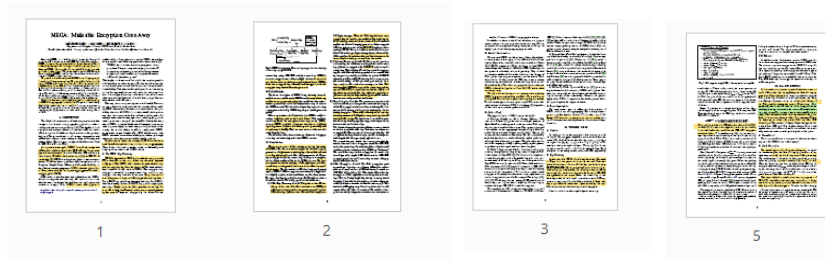
We only worked with our own test account when exploring MEGA's services and building our PoCs. We avoided overloading MEGA with login requests when running our attacks. We did not attempt to reverse engineer any MEGA server-side



It is not clear from the information provided whether MEGA has fixed or patched the shortcomings mentioned in the paper "MEGA: Malleable Encryption Goes Awry."



Kryptographie



Was nehmen wir als Infos von Seite fünf mit?

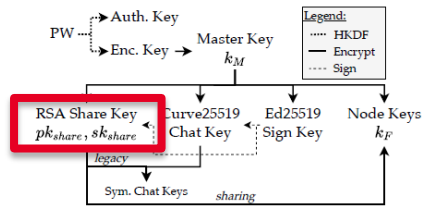


Fig. 1. MEGA's key hierarchy. The master key encrypts the share, chat, sign and node keys using AES-ECB.

$\text{DecSid}([sk_{share}^{encoded}]_{k_M}, [m]_{pk_{share}}):$

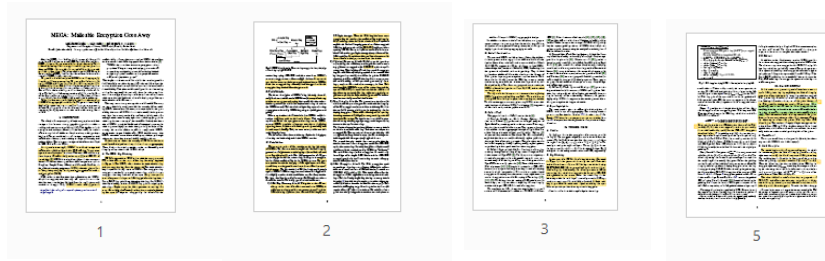
Given: encrypted RSA private key $[sk_{share}^{encoded}]_{k_M}$, encrypted message $[m]_{pk_{share}}$

Returns: decrypted and unpadded SID sid'

- 1 $sk_{share}^{encoded} \leftarrow \text{AES-ECB.Dec}(k_M, [sk_{share}^{encoded}]_{k_M})$
- 2 $N, e, d, p, q, d_p, d_q, u \leftarrow \text{DecodeRsaKey}(sk_{share}^{encoded})$
- 3 $m'_p \leftarrow ([m]_{pk_{share}})^{d_p} \bmod p$
- 4 $m'_q \leftarrow ([m]_{pk_{share}})^{d_q} \bmod q$
- 5 $t \leftarrow m'_p - m'_q \bmod p$
- 6 $h \leftarrow t \cdot u \bmod p$
- 7 $m' \leftarrow h \cdot q + m'_q$
- 8 $sid' \leftarrow m'[3:45] \parallel \text{Unpad 43 B SID.}$
- 9 **return** sid'

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

Kryptographie



Was nehmen wir als Infos von Seite fünf mit?

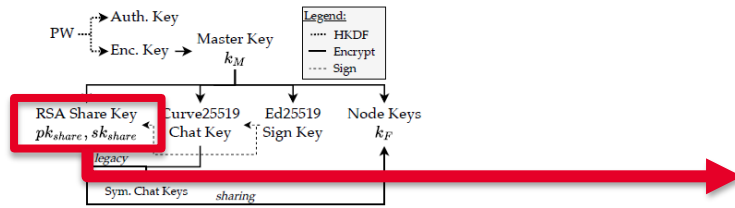


Fig. 1. MEGA's key hierarchy. The master key encrypts the share, chat, sign and node keys using AES-ECB.

$\text{DecSid}([sk_{share}^{encoded}]_{k_M}, [m]_{pk_{share}}):$

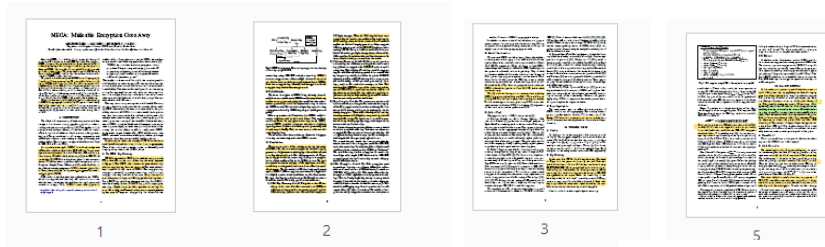
Given: encrypted RSA private key $[sk_{share}^{encoded}]_{k_M}$, encrypted message $[m]_{pk_{share}}$

Returns: decrypted and unpadded SID sid'

- 1 $sk_{share}^{encoded} \leftarrow \text{AES-ECB.Dec}(k_M, [sk_{share}^{encoded}]_{k_M})$
- 2 $N, e, d, p, q, d_p, d_q, u \leftarrow \text{DecodeRsaKey}(sk_{share}^{encoded})$
- 3 $m'_p \leftarrow ([m]_{pk_{share}})^{d_p} \bmod p$
- 4 $m'_q \leftarrow ([m]_{pk_{share}})^{d_q} \bmod q$
- 5 $t \leftarrow m'_p - m'_q \bmod p$
- 6 $h \leftarrow t \cdot u \bmod p$
- 7 $m' \leftarrow h \cdot q + m'_q$
- 8 $sid' \leftarrow m'[3:45] \parallel \text{Unpad 43 B SID.}$
- 9 **return** sid'

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

Kryptographie



Was nehmen wir als Infos von Seite fünf mit?

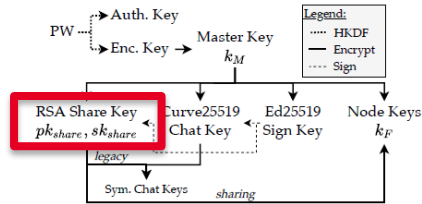


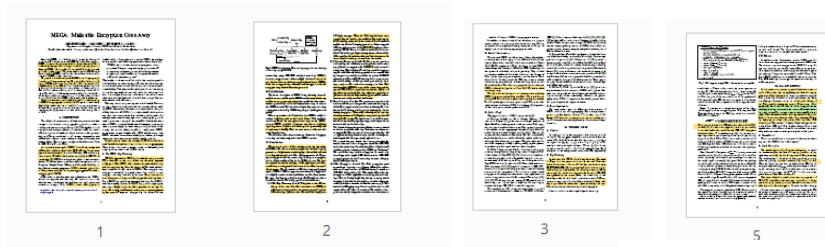
Fig. 1. MEGA's key hierarchy. The master key encrypts the share, chat, sign and node keys using AES-ECB.

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 $l(x)$ denotes the two-

```
DecSid([sk_share^encod]^k_M, [m]^pk_share);
Given: encrypted RSA private key [sk_share^encod]^k_M, encrypted
message [m]^pk_share
RETURN: decrypted and unpadding SID sid
1 sk_share^encod ← AES-ECB.Dec(k_M, [sk_share^encod]^k_M)
2 N, e, d, p, q, d_p, d_q, u ← DecodeRsaKey(sk_share^encod)
3 m_p ← ([m]^pk_share)^d mod p
4 m_q ← ([m]^pk_share)^d mod q
5 t ← m_p - m_q mod p
6 h ← t · u mod p
7 m' ← h · q + m_q
8 sid' ← m' [3:45] // Unpad 43 B SID.
9 return sid'
```

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

Kryptographie



Was nehmen wir als Infos von Seite fünf mit?

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 $l(x)$ denotes the two-

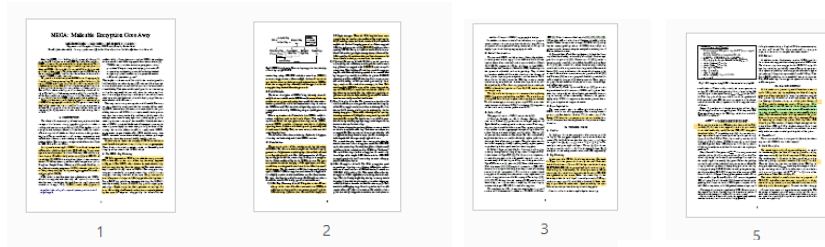
into leaking information about one of the prime factors of the RSA modulus during the session ID exchange. More

Fig. 1. MEGA's key hierarchy. The master key encrypts the share, chat, sign and node keys using AES-ECB.

```
DecSid([sk_share^encod]^k_M, [m]^pk_share);
Given: encrypted RSA private key [sk_share^encod]^k_M, encrypted
message [m]^pk_share
RETURN: decrypted and unpadding SID sid
1 sk_share^encod ← AES-ECB.Dec(k_M, [sk_share^encod]^k_M)
2 N, e, d, p, q, d_p, d_q, u ← DecodeRsaKey(sk_share^encod)
3 m_p ← ([m]^pk_share)^d mod p
4 m_q ← ([m]^pk_share)^d mod q
5 t ← m_p - m_q mod p
6 h ← t · u mod p
7 m' ← h · q + m_q
8 sid' ← m' [3:45] // Unpad 43 B SID.
9 return sid'
```

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

Kryptographie



Was nehmen wir als Infos von Seite fünf mit?

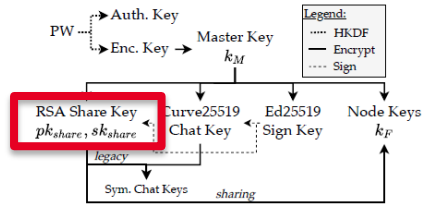


Fig. 1. MEGA's key hierarchy. The master key encrypts the share, chat, sign and node keys using AES-ECB.

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 $l(x)$ denotes the two-

into leaking information about one of the prime factors of the RSA modulus during the session ID exchange. More

the RSA modulus during the session ID exchange. More specifically, the session ID that the client decrypts with the mauled private key and sends to the server will reveal whether the prime is smaller or greater than an adversarially-chosen value. This information enables a binary search for the prime

```
DecSid([sk_share^encod]^k_M, [m]^pk_share);
Given: encrypted RSA private key [sk_share^encod], encrypted
message [m]
1 sk_share^encod ← AES-ECB.Dec(k_M, [sk_share^encod]^k_M)
2 N, e, d, p, q, d_p, d_q, u ← DecodeRsaKey(sk_share^encod)
3 m_p ← ([m]^pk_share)^d mod p
4 m_q ← ([m]^pk_share)^d mod q
5 t ← m_p - m_q mod p
6 h ← t · u mod p
7 m' ← h · q + m_q
8 sid' ← m'[3:45] // Unpad 43 B SID.
9 return sid'
```

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

Kryptographie

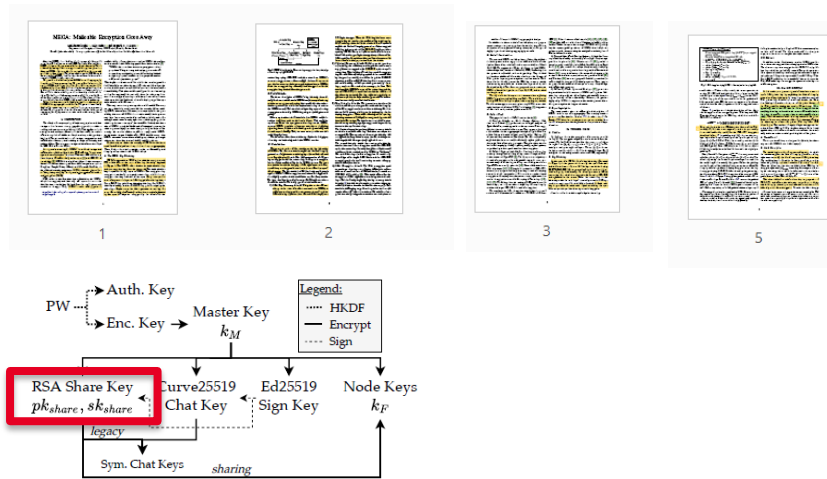


Fig. 1. MEGA's key hierarchy. The master key encrypts the share, chat, sign and node keys using AES-ECB.

```
DecSid([sk_share^encoded]_k_M, [m]_pk_share);
Given: encrypted RSA private key [sk_share^encoded]_k_M, encrypted
message [m]_pk_share
RETURN: decrypted and unpadding SID sid
1 sk_share^encoded ← AES-ECB.Dec(k_M, [sk_share^encoded]_k_M)
2 N, e, d, p, q, d_p, d_q, u ← DecodeRsaKey(sk_share^encoded)
3 m_p ← ([m]_pk_share)^d_q mod p
4 m_q ← ([m]_pk_share)^d_p mod q
5 t ← m_p - m_q mod p
6 h ← t · u mod p
7 m' ← h · q + m_q
8 sid' ← m'[3:45] // Unpad 43 B SID.
9 return sid'
```

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

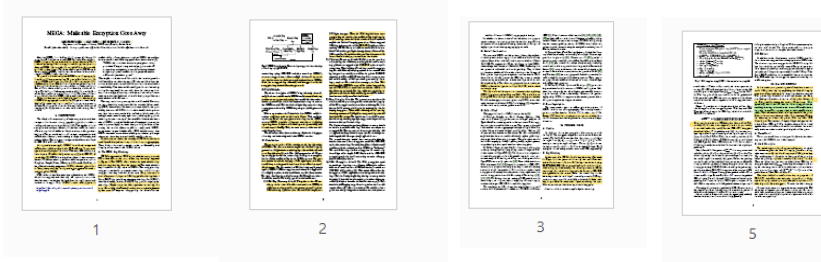
Here, q and p are the two 1024-bit prime factors of the RSA modulus N , d is the secret exponent, and $u \leftarrow q^{-1} \bmod p$ is an additional value useful for the RSA-CRT decryption described below. P is padding and $l(x)$ denotes the two-

into leaking information about one of the prime factors of the RSA modulus during the session ID exchange. More

the RSA modulus during the session ID exchange. More specifically, the session ID that the client decrypts with the mauled private key and sends to the server will reveal whether the prime is smaller or greater than an adversarially-chosen value. This information enables a binary search for the prime

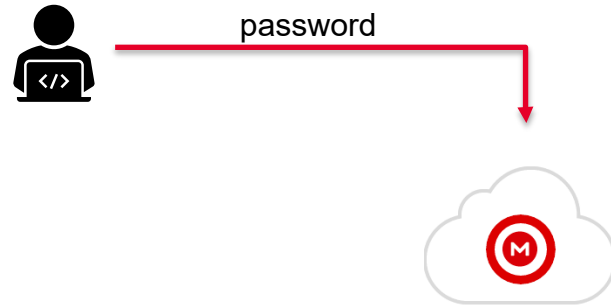
value. This information enables a binary search for the prime factor, with one comparison per client login attempt, allowing the adversary to recover the private RSA key after 1023 client logins. The number of required login attempts can be

Kryptographie



into leaking information about one of the prime factors of the RSA modulus during the session ID exchange. More

Wir müssen also Fig. 5 verstehen.



DecSid($[sk_{share}^{encoded}]_{k_M}, [m]_{pk_{share}}$):

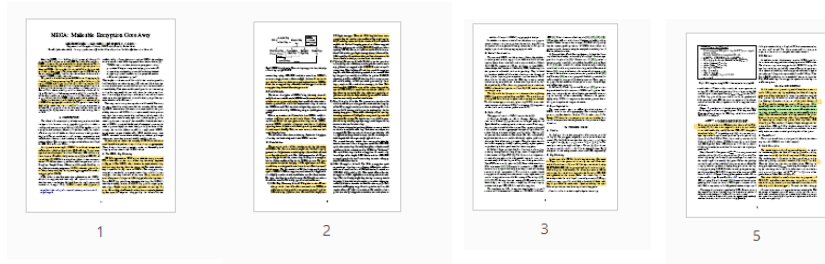
Given: encrypted RSA private key $[sk_{share}^{encoded}]_{k_M}$, encrypted message $[m]_{pk_{share}}$

Returns: decrypted and unpadded SID sid'

- 1 $sk_{share}^{encoded} \leftarrow \text{AES-ECB.Dec}(k_M, [sk_{share}^{encoded}]_{k_M})$
- 2 $N, e, d, p, q, d_p, d_q, u \leftarrow \text{DecodeRsaKey}(sk_{share}^{encoded})$
- 3 $m'_p \leftarrow ([m]_{pk_{share}})^{d_p} \bmod p$
- 4 $m'_q \leftarrow ([m]_{pk_{share}})^{d_q} \bmod q$
- 5 $t \leftarrow m'_p - m'_q \bmod p$
- 6 $h \leftarrow t \cdot u \bmod p$
- 7 $m' \leftarrow h \cdot q + m'_q$
- 8 $sid' \leftarrow m'[3:45] \parallel \text{Unpad 43 B SID.}$
- 9 **return** sid'

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

Kryptographie



into leaking information about one of the prime factors of the RSA modulus during the session ID exchange. More

Wir müssen also Fig. 5 verstehen.

```
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  $sid'$   
    1  $sk_{share}^{encoded} \leftarrow \text{AES-ECB.Dec}(k_M, [sk_{share}^{encoded}]_{k_M})$   
    2  $N, e, d, p, q, d_p, d_q, u \leftarrow \text{DecodeRsaKey}(sk_{share}^{encoded})$   
    3  $m'_p \leftarrow ([m]_{pk_{share}})^{d_p} \bmod p$   
    4  $m'_q \leftarrow ([m]_{pk_{share}})^{d_q} \bmod q$   
    5  $t \leftarrow m'_p - m'_q \bmod p$   
    6  $h \leftarrow t \cdot u \bmod p$   
    7  $m' \leftarrow h \cdot q + m'_q$   
    8  $sid' \leftarrow m'[3:45] \parallel \text{Unpad 43 B SID.}$   
    9 return  $sid'$ 
```

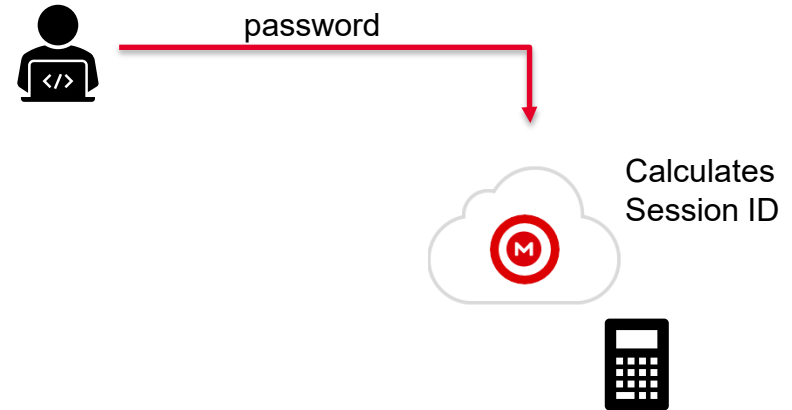
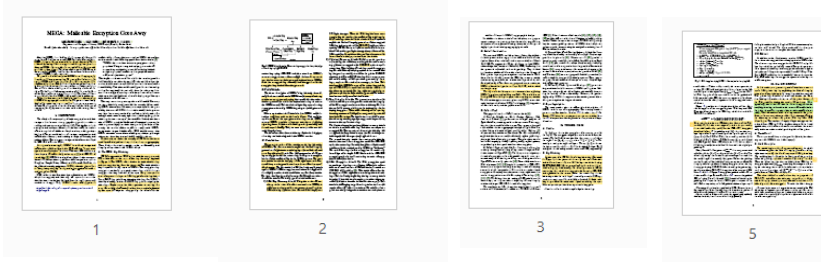


Fig. 5. SID decryption during MEGA's client authentication using RSA.

Kryptographie



into leaking information about one of the prime factors of the RSA modulus during the session ID exchange. More

Wir müssen also Fig. 5 verstehen.

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'

- $sk_{share}^{encoded} \leftarrow \text{AES-ECB.Dec}(k_M, [sk_{share}^{encoded}]_{k_M})$
- $N, e, d, p, q, d_p, d_q, u \leftarrow \text{DecodeRsaKey}(sk_{share}^{encoded})$
- $m'_p \leftarrow ([m]_{pk_{share}})^{d_p} \bmod p$
- $m'_q \leftarrow ([m]_{pk_{share}})^{d_q} \bmod q$
- $t \leftarrow m'_p - m'_q \bmod p$
- $h \leftarrow t \cdot u \bmod p$
- $m' \leftarrow h \cdot q + m'_q$
- $sid' \leftarrow m'[3:45] \parallel \text{Unpad 43 B SID.}$
- return** sid'

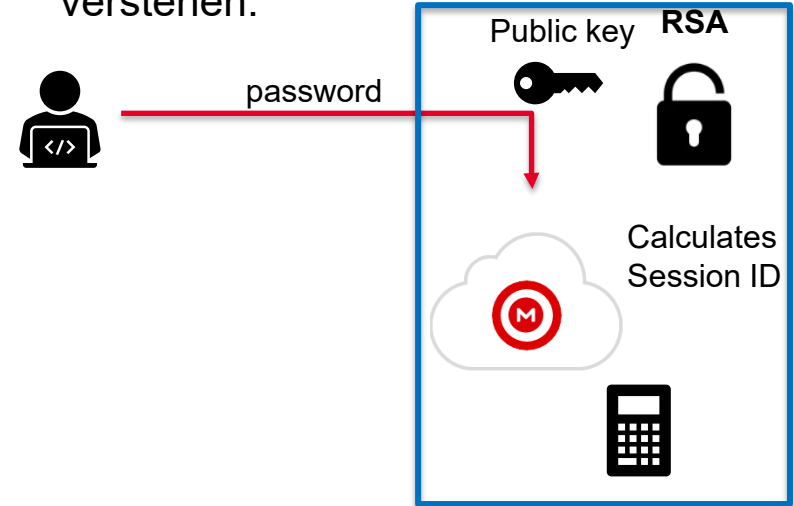
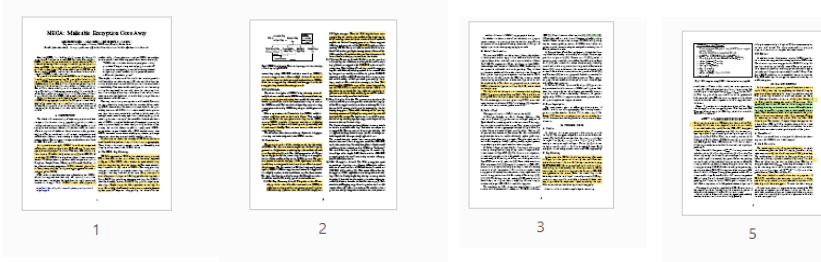


Fig. 5. SID decryption during MEGA's client authentication using RSA.

Kryptographie



into leaking information about one of the prime factors of the RSA modulus during the session ID exchange. More

Wir müssen also Fig. 5 verstehen.

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'

- $sk_{share}^{encoded} \leftarrow \text{AES-ECB.Dec}(k_M, [sk_{share}^{encoded}]_{k_M})$
- $N, e, d, p, q, d_p, d_q, u \leftarrow \text{DecodeRsaKey}(sk_{share}^{encoded})$
- $m'_p \leftarrow ([m]_{pk_{share}})^{d_p} \bmod p$
- $m'_q \leftarrow ([m]_{pk_{share}})^{d_q} \bmod q$
- $t \leftarrow m'_p - m'_q \bmod p$
- $h \leftarrow t \cdot u \bmod p$
- $m' \leftarrow h \cdot q + m'_q$
- $sid' \leftarrow m'[3:45] \parallel \text{Unpad 43 B SID.}$
- return** sid'

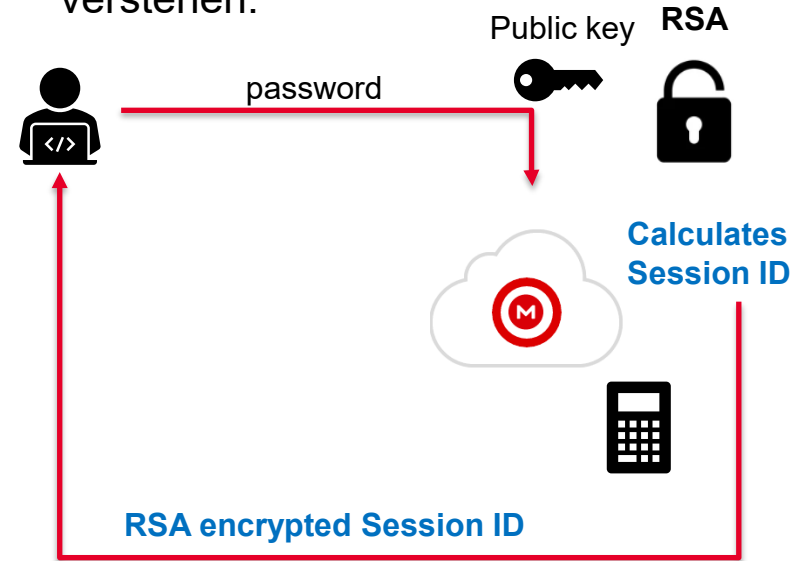
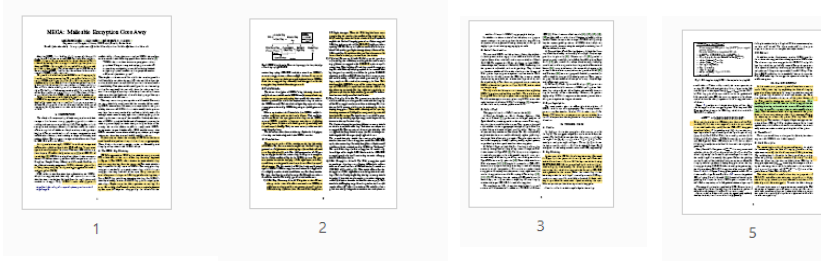


Fig. 5. SID decryption during MEGA's client authentication using RSA.

Kryptographie



into leaking information about one of the prime factors of the RSA modulus during the session ID exchange. More

Wir müssen also Fig. 5 verstehen.

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'

- $sk_{share}^{encoded} \leftarrow \text{AES-ECB.Dec}(k_M, [sk_{share}^{encoded}]_{k_M})$
- $N, e, d, p, q, d_p, d_q, u \leftarrow \text{DecodeRsaKey}(sk_{share}^{encoded})$
- $m'_p \leftarrow ([m]_{pk_{share}})^{d_p} \bmod p$
- $m'_q \leftarrow ([m]_{pk_{share}})^{d_q} \bmod q$
- $t \leftarrow m'_p - m'_q \bmod p$
- $h \leftarrow t \cdot u \bmod p$
- $m' \leftarrow h \cdot q + m'_q$
- $sid' \leftarrow m'[3:45] \parallel \text{Unpad 43 B SID.}$
- return** sid'

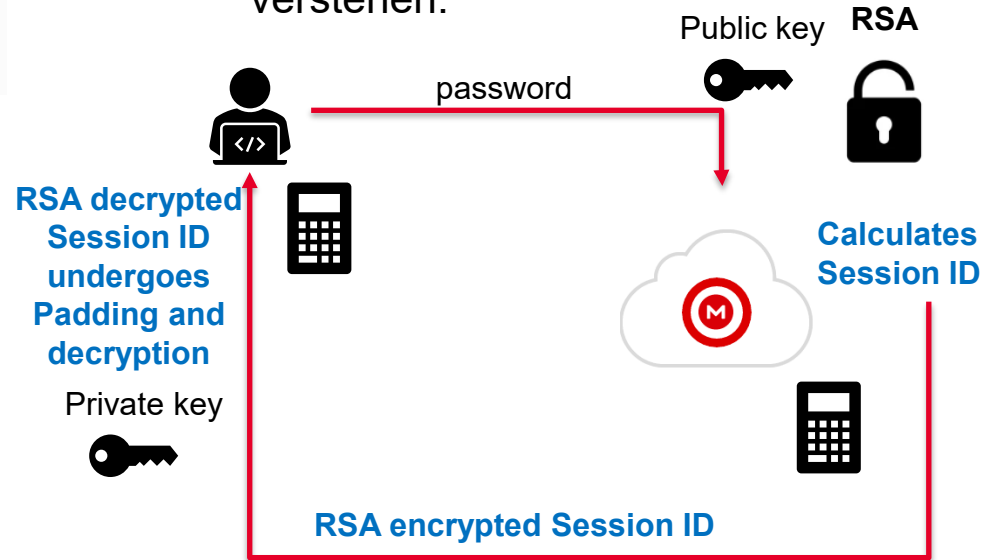
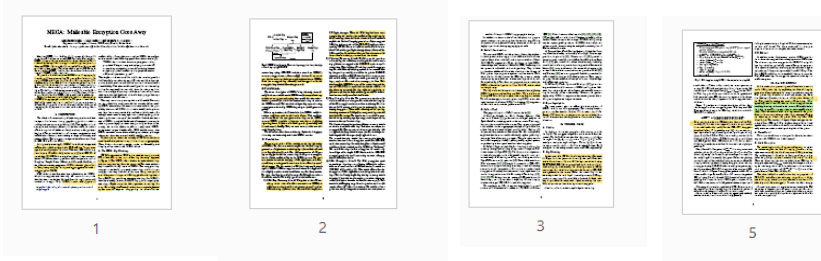


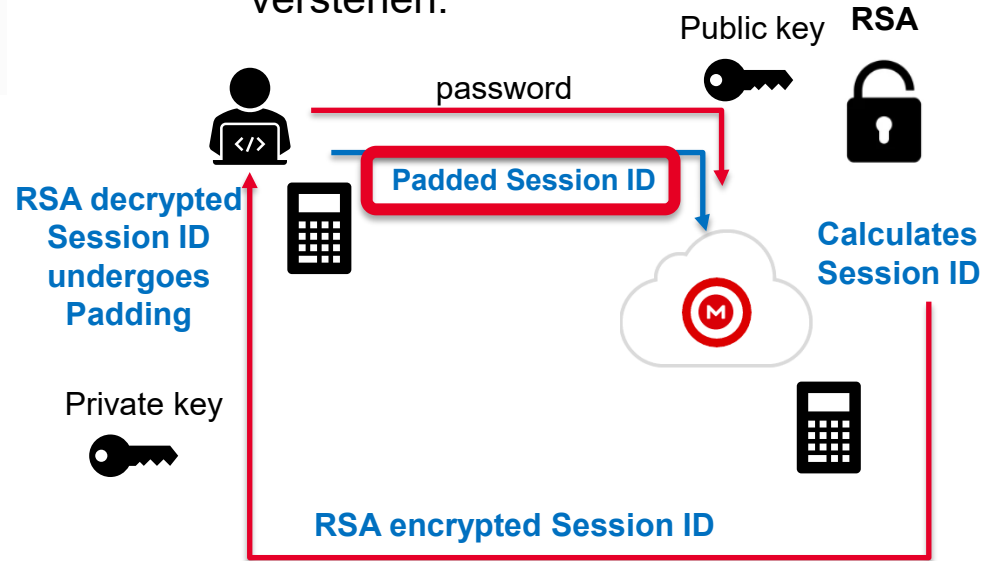
Fig. 5. SID decryption during MEGA's client authentication using RSA.

Kryptographie



into leaking information about one of the prime factors of the RSA modulus during the session ID exchange. More

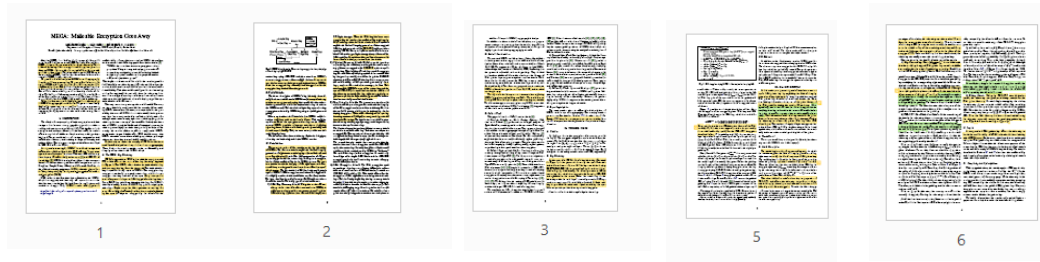
Wir müssen also Fig. 5 verstehen.



```
DecSid([sk_share^encoded]_{k_M}, [m]_{pk_share}):
    Given: encrypted RSA private key [sk_share^encoded]_{k_M}, encrypted
    message [m].
    Returns: decrypted and unpadding SID sid'
1 sk_share^encoded ← AES-ECB.Dec(k_M, [sk_share^encoded]_{k_M})
2 N, e, d, p, q, d_p, d_q, u ← DecodeRsaKey(sk_share^encoded)
3 m'_p ← ([m]_{pk_share})^{d_p} mod p
4 m'_q ← ([m]_{pk_share})^{d_q} mod q
5 t ← m'_p - m'_q mod p
6 h ← t · u mod p
7 m' ← h · q + m'_q
8 sid' ← m'[3:45] // Unpad 43 B SID.
9 return sid'
```

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

Kryptographie



Was nehmen wir als Infos von Seite sechs mit?

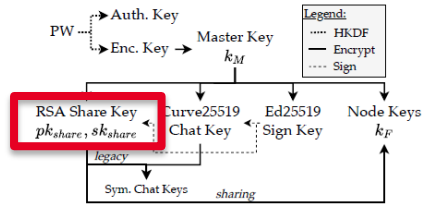


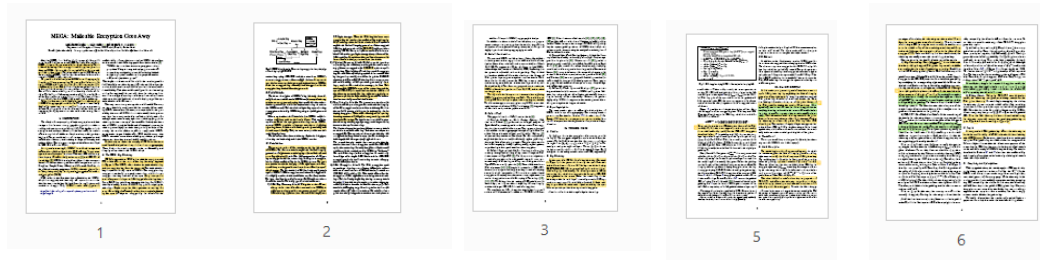
Fig. 1. MEGA's key hierarchy. The master key encrypts the share, chat, sign and node keys using AES-ECB.

```

DecSid([sk_share^encoded]_k_M, [m]_pk_share);
Given: encrypted RSA private key [sk_share^encoded]_k_M, encrypted
message [m]_pk_share.
RETURN: decrypted and unpadding SID sid.
1 sk_share^encoded ← AES-ECB.Dec(k_M, [sk_share^encoded]_k_M)
2 N, e, d, p, q, d_p, d_q, u ← DecodeRsaKey(sk_share^encoded)
3 m_p ← ([m]_pk_share)^d mod p
4 m_q ← ([m]_pk_share)^d mod q
5 t ← m_p - m_q mod p
6 h ← t · u mod p
7 m' ← h · q + m_q
8 sid' ← m' [3:45] // Unpad 43 B SID.
9 return sid'
  
```

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

Kryptographie



Was nehmen wir als Infos von Seite sechs mit?

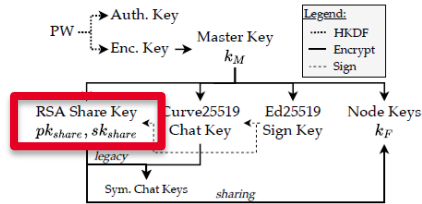


Fig. 1. MEGA's key hierarchy. The master key encrypts the share, chat, sign and node keys using AES-ECB.

B. Attack Description

The attack begins with a *key overwriting* step, in which the attacker exploits the lack of integrity protection of key ciphertexts to modify the client's outsourced RSA private key. The resulting key is altered in the last part of the encoding before the padding, such that it contains $u' \neq u = q^{-1} \bmod p$. When the client uses the thus modified private RSA key to decrypt a ciphertext $[m]_{pk_{share}}$ containing a message m chosen by the adversary, the result leaks information about whether $m < q$ or $m \geq q$, where $q \in [2^{1023}, 2^{1024} - 1]$ is one of the prime factors of the RSA modulus N .

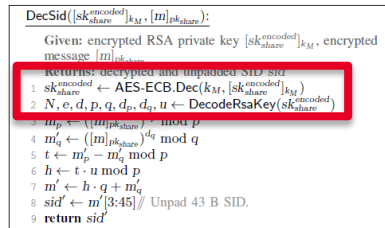
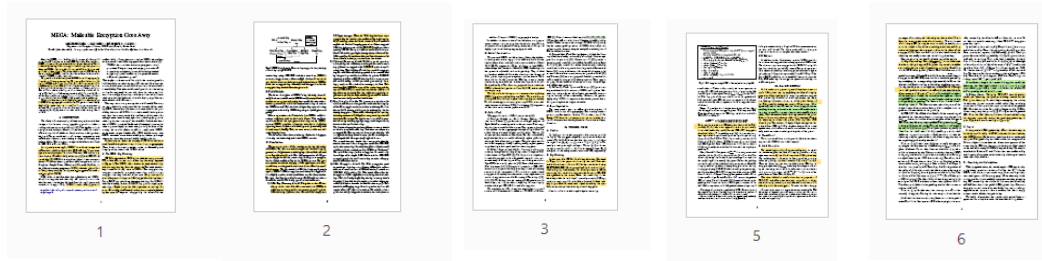


Fig. 5. SID decryption during MEGA's client authentication using RSA.

Kryptographie



Was nehmen wir als
Infos von Seite sechs
mit?



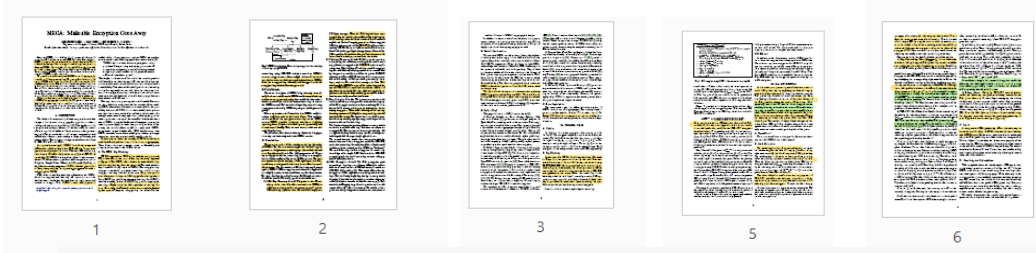
Übersetze und Vereinfache.



B. Attack Description

The attack begins with a key overwriting step, in which the attacker exploits the lack of integrity protection of key ciphertexts to modify the client's outsourced RSA private key. The resulting key is altered in the last part of the encoding before the padding, such that it contains $u06 = u = q \cdot 1 \bmod p$. When the client uses the thus modified private RSA key to decrypt a ciphertext $[m]_{pkshare}$ containing a message m chosen by the adversary, the result leaks information about whether $m < q$ or $m \geq q$, where $q \in [2^{1023}; 2^{1024} - 1]$ is one of the prime factors of the RSA modulus N .

Kryptographie



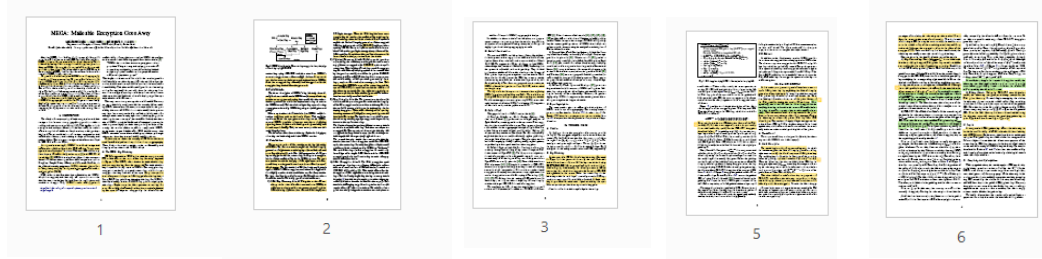
Was nehmen wir als Infos von Seite sechs mit?



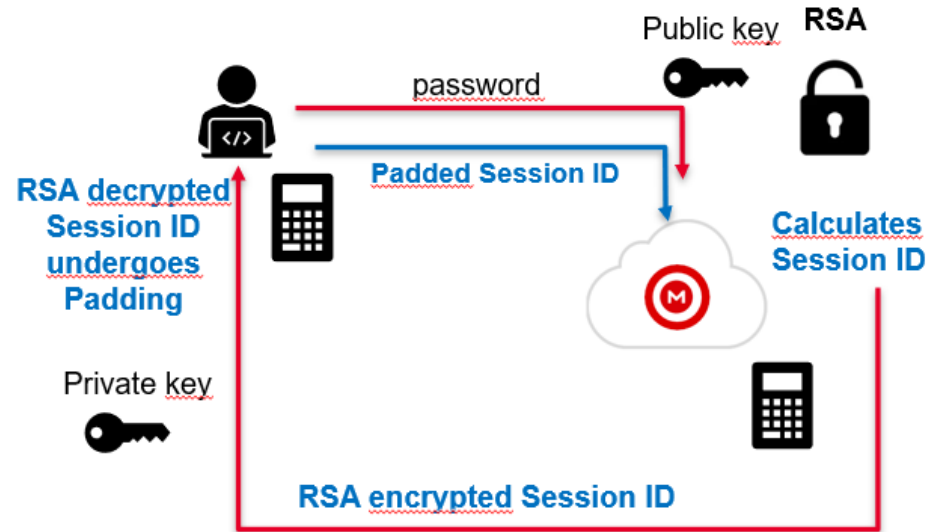
Die Attacke beginnt mit einem sogenannten "Key Overwriting", bei dem der Angreifer das Fehlen von Integritäts-Schutz von Verschlüsselungscodes für den privaten RSA-Schlüssel des Clients ausnutzt, um diesen zu modifizieren. Der so modifizierte Schlüssel wird in einem Teil der Verschlüsselung vor dem Padding verändert, sodass er einen u0-Wert enthält, der ungleich dem ursprünglichen u-Wert ($u = q - 1 \bmod p$) ist. Wenn der Client den so modifizierten privaten RSA-Schlüssel verwendet, um einen Ciphertext [m]pkshare zu entschlüsseln, der eine vom Angreifer ausgewählte Nachricht m enthält, gibt das Ergebnis Informationen darüber preis, ob m kleiner oder größer als q ist, wobei q zu den Primfaktoren von N (dem RSA-Modulus) gehört und im Bereich von 21023 bis 21024 - 1 liegt.



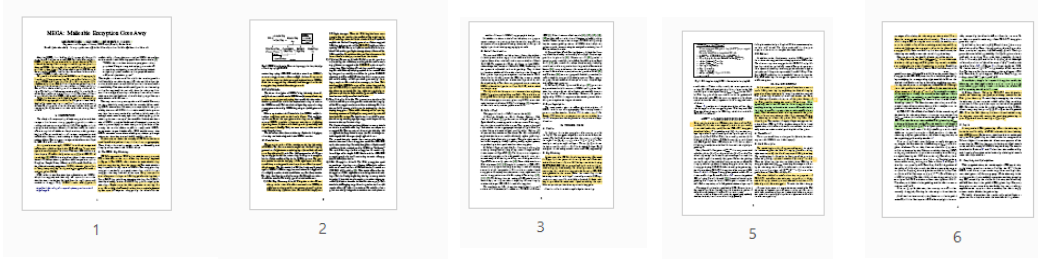
Kryptographie



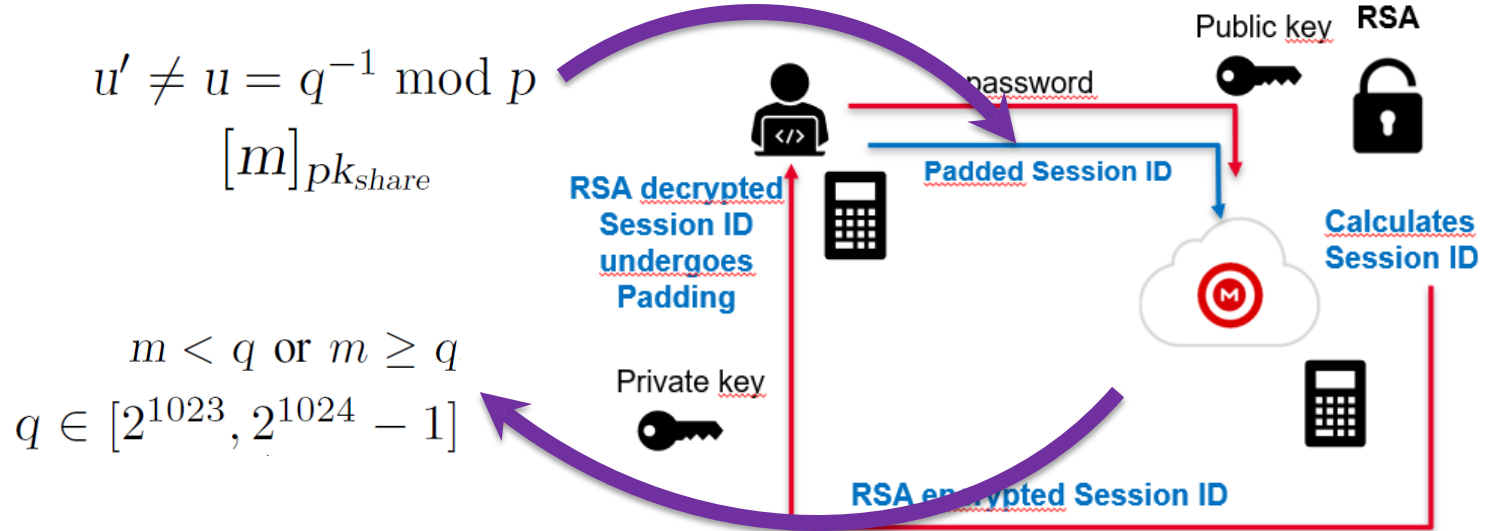
Was nehmen wir als
Infos von Seite sechs
mit?



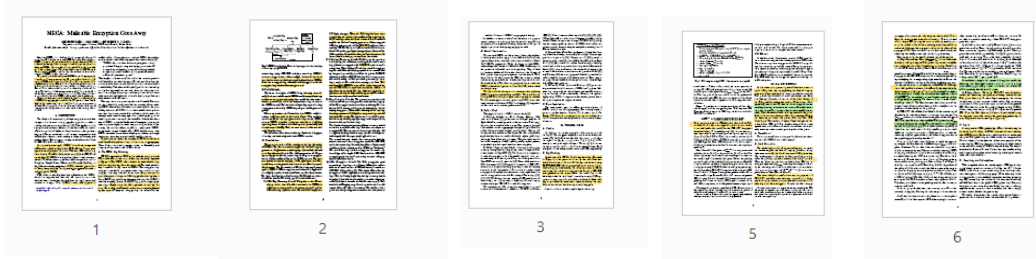
Kryptographie



Was nehmen wir als
Infos von Seite sechs
mit?



Kryptographie



Was nehmen wir als Infos von Seite sechs mit?

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'

- $sk_{share}^{encoded} \leftarrow \text{AES-ECB.Dec}(k_M, [sk_{share}^{encoded}]_{k_M})$
- $N, e, d, p, q, d_p, d_q, u \leftarrow \text{DecodeRsaKey}(sk_{share}^{encoded})$
- $m'_p \leftarrow ([m]_{pk_{share}})^{d_p} \bmod p$
- $m'_q \leftarrow ([m]_{pk_{share}})^{d_q} \bmod q$
- $t \leftarrow m'_p - m'_q \bmod p$
- $h \leftarrow t \cdot u \bmod p$
- $m' \leftarrow h \cdot q + m'_q$
- $sid' \leftarrow m'[3:45] \parallel \text{Unpad 43 B SID.}$
- return** sid'

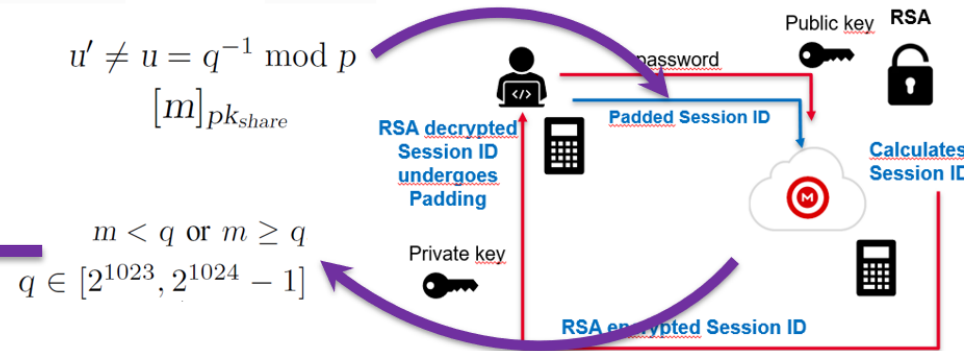
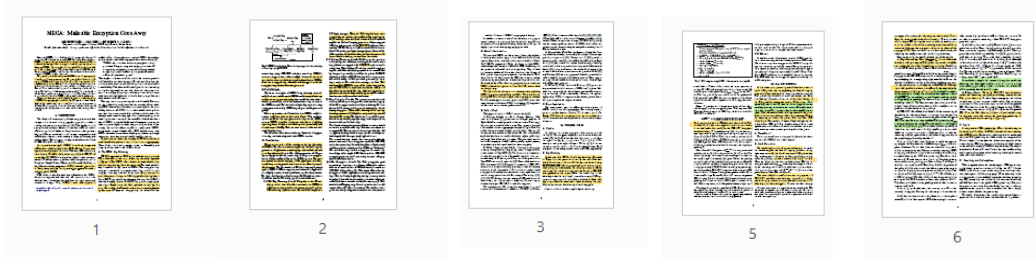


Fig. 5. SID decryption during MEGA's client authentication using RSA.

Kryptographie



Wiederholen bis wir in Besitz von q sind.

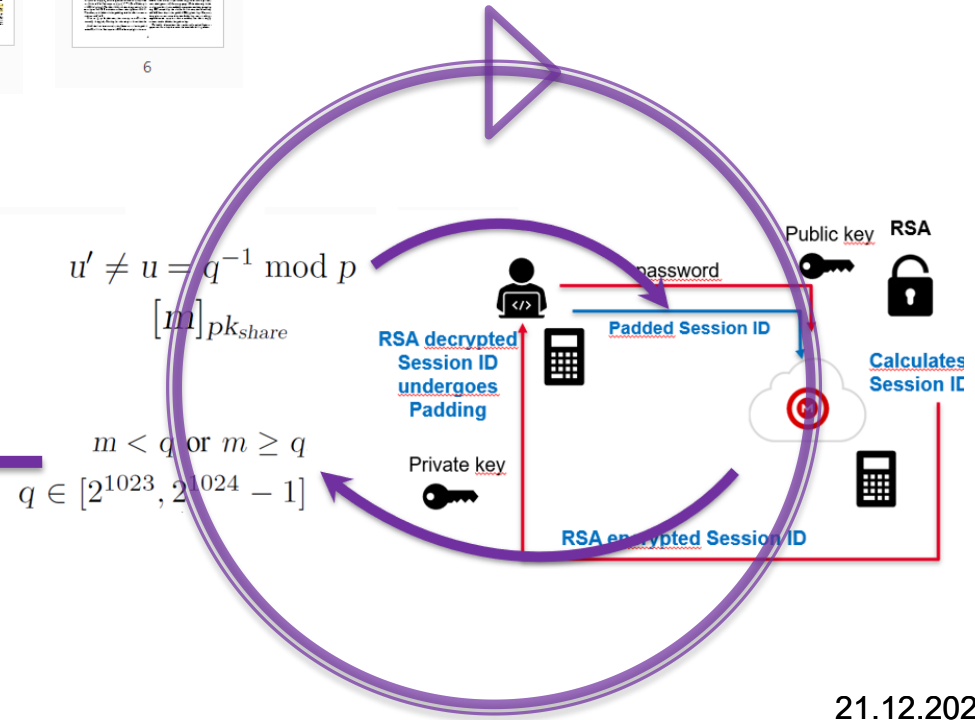
DecSid $([sk_{share}^{encoded}]_{k_M}, [m]_{pk_{share}})$:

Given: encrypted RSA private key $[sk_{share}^{encoded}]_{k_M}$, encrypted message $[m]_{pk_{share}}$

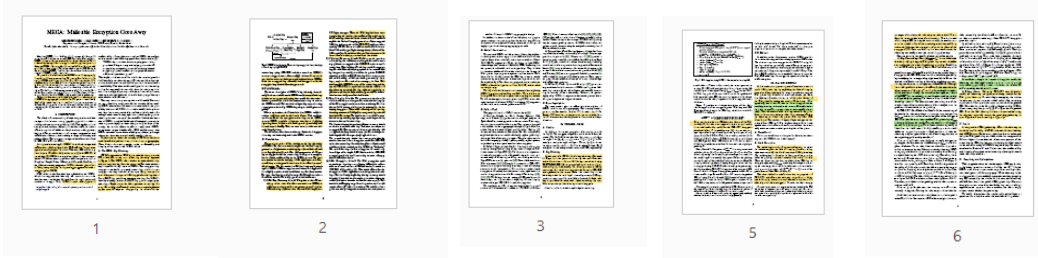
Returns: decrypted and unpadded SID sid'

- $sk_{share}^{encoded} \leftarrow \text{AES-ECB.Dec}(k_M, [sk_{share}^{encoded}]_{k_M})$
- $N, e, d, p, q, d_p, d_q, u \leftarrow \text{DecodeRsaKey}(sk_{share}^{encoded})$
- $m'_p \leftarrow ([m]_{pk_{share}})^{d_p} \bmod p$
- $m'_q \leftarrow ([m]_{pk_{share}})^{d_q} \bmod q$
- $t \leftarrow m'_p - m'_q \bmod p$
- $h \leftarrow t \cdot u \bmod p$
- $m' \leftarrow h \cdot q + m'_q$
- $sid' \leftarrow m'[3:45] \parallel \text{Unpad 43 B SID.}$
- return** sid'

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



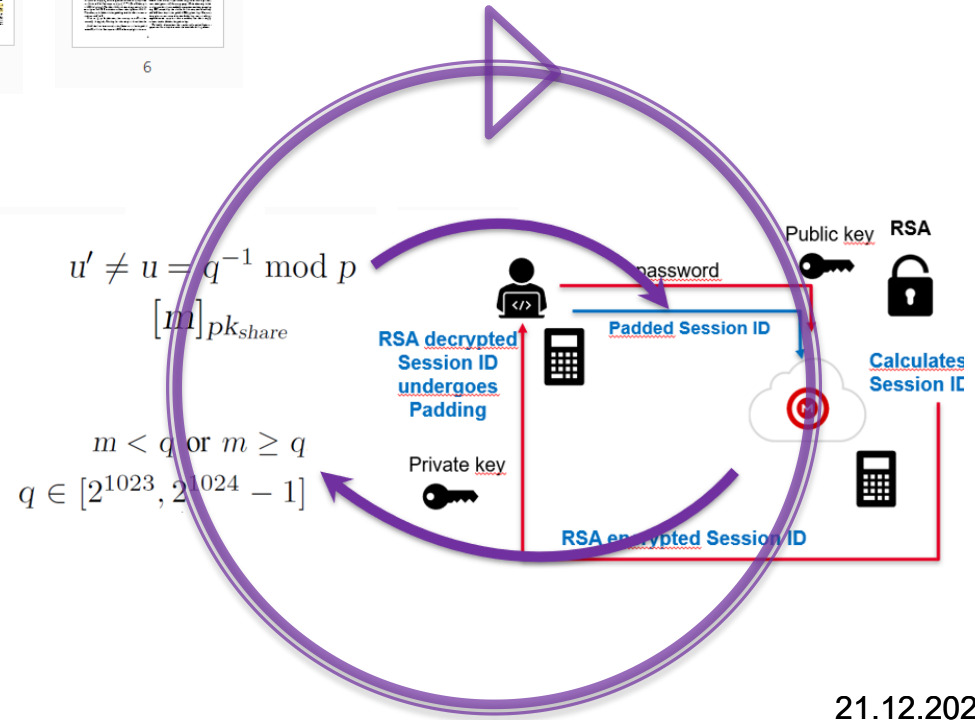
Kryptographie



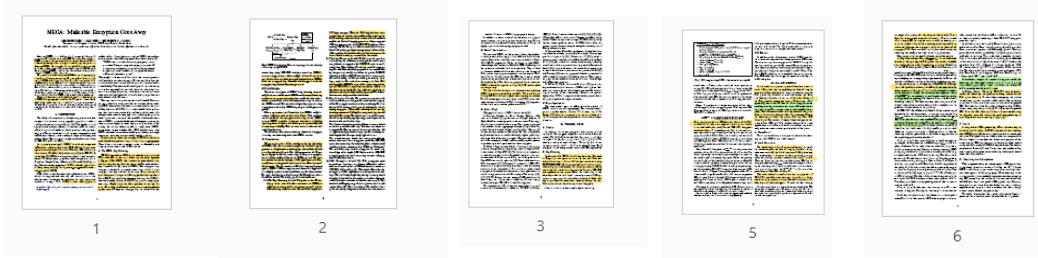
Wiederholen bis wir in Besitz von q sind.

Warum ist das möglich?

Next, `DecodeRsaKey` parses $sk_{share}^{encoded}$ into components and calculates $d_p \leftarrow (d-1) \bmod p$ and $d_q \leftarrow (d-1) \bmod q$. The client only sanity checks the length encodings to ensure that the result is split into exactly four parts. Neither the padding nor the lengths of the individual components are verified. No integrity check is performed during decryption since the key is encrypted using AES-ECB. After decoding the private key,



Kryptographie

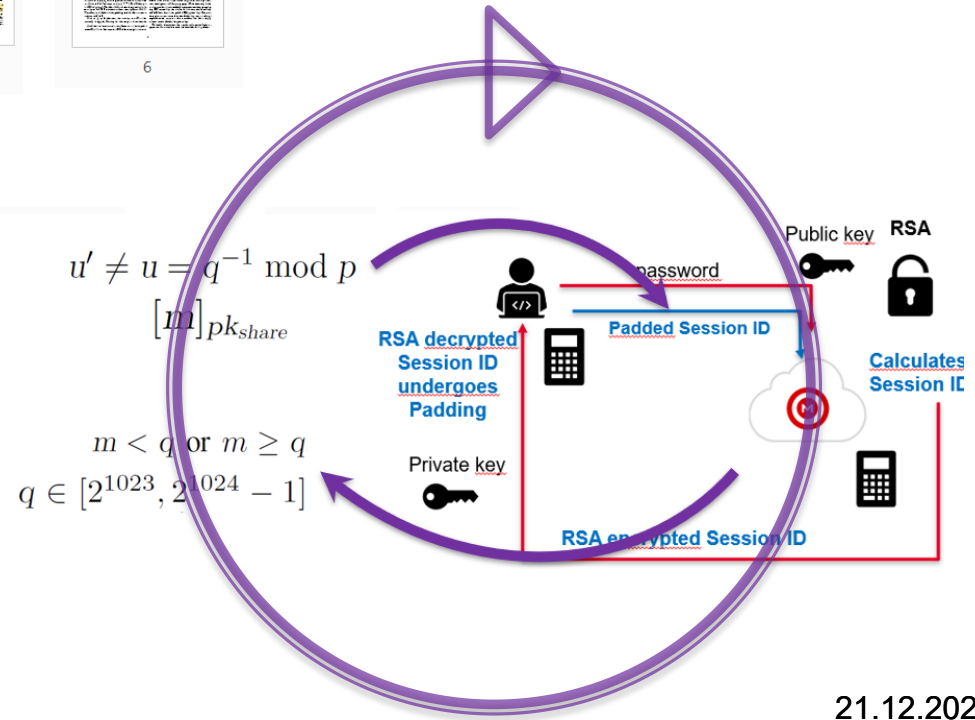


Was wird empfohlen?

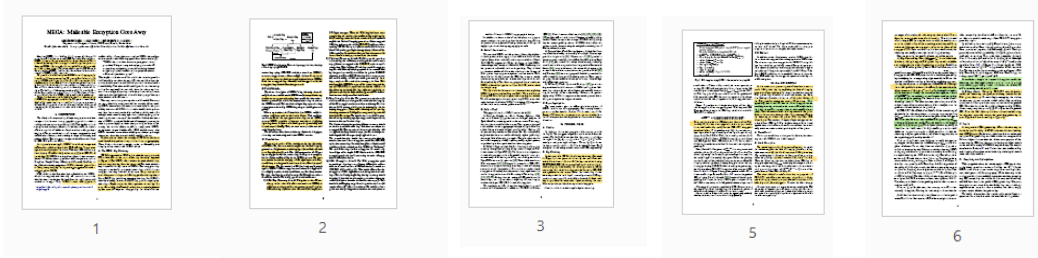
A. Immediate Countermeasures

1) *Integrity-Protect Key Ciphertexts*: The most effective countermeasure against our attacks is to add integrity protection for the encrypted user keys stored by MEGA. This can be

Wiederholen bis wir in Besitz von q sind.



Kryptographie



Was wird empfohlen?

A. Immediate Countermeasures

1) *Integrity-Protect Key Ciphertexts*: The most effective countermeasure against our attacks is to add integrity protection for the encrypted user keys stored by MEGA. This can be

Wiederholen bis wir in Besitz von q sind.

