Encrypted MultiChannel Communication (EMC^2): Johnny Should Use Secret Sharing

Gowri R. Chandran
Technical University of Darmstadt
Darmstadt, Germany
chandran@encrypto.tu-darmstadt.de

Kilian Demuth
Technical University of Darmstadt
Darmstadt, Germany
demuth@peasec.tu-darmstadt.de

Kasra Edalatnejad*
Technical University of Darmstadt
Darmstadt, Germany
edalat@encrypto.tu-darmstadt.de

Sebastian Linsner
Technical University of Darmstadt
Darmstadt, Germany
linsner@peasec.tu-darmstadt.de

Christian Reuter
Technical University of Darmstadt
Darmstadt, Germany
reuter@peasec.tu-darmstadt.de

Thomas Schneider
Technical University of Darmstadt
Darmstadt, Germany
schneider@encrypto.tu-darmstadt.de

Abstract

Nowadays, the problem of point-to-point encryption is solved by the wide adaptation of protocols like TLS. However, challenges persist for End-to-End Encryption (E2EE). Current E2EE solutions, such as PGP and secure messengers like Signal, suffer from issues like 1) low usability, 2) small user base, 3) dependence on central service providers, and 4) susceptibility to backdoors. Concerns over legally mandated backdoors are rising as the US and EU are proposing new surveillance regulations requiring chat monitoring. We present a new E2EE solution called Encrypted MultiChannel Communication (EMC^2), based on n-out-of-n secret sharing. EMC^2 splits messages into multiple secret shares and sends them through independent channels. We show that multiple independent channels exist between users and *EMC^2 provides E2EE with no single point of trust, no setup, and is understandable by the general public.* Our solution complements existing tools and strengthens the case against legally enforced backdoors by demonstrating their ineffectiveness.

CCS Concepts

• **Security and privacy** → *Information-theoretic techniques.*

Keywords

end-to-end encryption; secret sharing; OTP

ACM Reference Format:
Gowri R. Chandran, Kilian Demuth, Kasra Edalatnejad, Sebastian Linsner, Christian Reuter, and Thomas Schneider. 2024. EMC^2: Johnny Should Use Secret Sharing. In *Proceedings of the 23rd Workshop on Privacy in the Electronic Society (WPES '24), October 14–18, 2024, Salt Lake City, UT, USA.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3689943.3695051

1 Introduction

Most Internet communication today occurs through encrypted channels. While the adoption of protocols such as TLS [30, 38] has resolved point-to-point encryption, practical challenges remain for E2EE. The primary E2EE solutions are encrypted emails using PGP [20] and S/MIME [41], and secure messengers like Signal [43]. Solutions based on PGP require users to manage cryptographic keys and certificates. These solutions suffer from severe usability issues [40, 56] and low adaption rates. Secure messengers based on protocols like Signal [1, 49] improve usability by making key management transparent. However, they still face challenges like requiring trust in central service providers especially in proprietary applications with closed-source code like WhatsApp, whereas open-source alternatives like Signal have limited user bases. Messengers are deployed by service providers that face legal threats from proposed surveillance regulations in the US [53] and the EU [19]. These regulations aim to mandate monitoring and backdoors in encrypted communication to assist law enforcement or to target child sexual abuse material. Academics have warned about the dangers these laws pose to E2EE, democracy, and fundamental rights [50]. Despite academic efforts, requests for unrealistic solutions from policymakers persist due to a lack of technical understanding.

We take a different perspective to achieving E2EE, aiming to tackle trust and regulation challenges. Our core idea is very simple. We split messages into multiple secret shares and send them through independent channels where all but one channel can be broken/backdoored. We prioritize the following goals in our solution:

Public understandability. The protocol should be understandable by the general public with no background in math or IT.

No single point of trust. The protocol should not depend on trusting a single service provider such that it is infeasible to mandate backdoors or client monitoring through a single provider.

No setup. To achieve high receiver coverage, users should be able to initiate secure communication without requiring recipients to generate public keys or register for a specific application.

While prior work offer elegant solutions for E2EE messaging, these solutions often rely on complex mathematical constructs inaccessible to the general public. We believe understandable solutions are essential for public and political debates. Forcing messengers to weaken E2EE by introducing backdoors negatively impacts the rights of all users. We demonstrate that these regulations will not hinder the capability of users, even those without technical background, to initiate confidential communications.

We design our E2EE solution based on n-out-of-n secret sharing which is a generalization of One-Time Pad (OTP) encryption.

*Contact author.

WPES '24, October 14–18, 2024, Salt Lake City, UT, USA

ACM ISBN 979-8-4007-1239-5/24/10
https://doi.org/10.1145/3689943.3695051

The OTP encryption, introduced in the early 20th century, is the first cipher to achieve information-theoretic security. It is one of the simplest ciphers, requiring only an XOR operation between the message and a random key. However, as it requires sending a random key that is as long as the message through an independent secure channel, OTP is rarely used in practice. Generating long random keys is easy nowadays using a pseudorandom generator (PRG) like AES in counter mode. We argue that today multiple independent and secure channels already exist between users. On average, users have two mail addresses and six social media/messenger accounts. Each communication channel is encrypted at least in a client-server fashion, where only the service providers can eavesdrop on the messages. We propose that combining multiple independent channels through OTP enables E2EE.

Roadmap. We design an E2EE messaging solution called Encrypted MultiChannel Communication based on OTP and study its properties in §2. We show that multiple independent secure channels exist between users in §3. We systematize existing private communication tools and compare them to EMC^2 in §4. Finally, we discuss the next steps and how EMC^2 improves the ecosystem in §5.

We do not envision EMC^2 as a replacement for general-purpose messengers like Signal. EMC^2 does not provide anonymity and users are often not motivated to take extra steps for E2EE in typical use cases. However, we believe solutions understandable by a general audience are essential for public debates. The lack of setup in EMC^2 allows contacting arbitrary users and achieving high receiver coverage. Its simplicity enables a secure, verifiable solution without public-key cryptography or trusted service providers, achievable in just a few lines of code. Secure E2EE messengers still face limitations such as reliance on central platforms and regulatory pressures. EMC^2 complements these solutions by enabling senders to distribute the trust across multiple service providers without requiring any server-side change. By demonstrating the ineffectiveness of laws mandating backdoors, EMC^2 aims to strengthen the arguments against such laws and alleviate pressure on server-side solutions.

2 Design of EMC^2

At a high level, there are three steps for sending messages via EMC^2. 1) We provide a tool that allows users to encrypt/decrypt a message M using OTP which only requires an XOR operation such that $M = m_1 \oplus m_2$. The simplicity and statelessness of this operation allow EMC^2 to run offline and be deployed as a program, script, or static webpage[1] 2) EMC^2 adds a plaintext preamble to encrypted message shares that explains how the decryption works. 3) The sender manually sends the two encrypted shares m_1 and m_2 to the receiver through two independent channels. We define communication channels, explain how EMC^2 works, and study its properties.

Channel. Communication occurs through various links such as an instant message, SMS, email, or social media direct message. We model a channel c, that supports two operations c.Send(M) and $M \leftarrow c$.Recv(), as a series of service providers that relay data from a sender to a receiver through point-to-point encrypted links. While P2P encryption prevents network observers from eavesdropping, service providers may access plaintext message M. We denote the set of providers observing the plain message as c.Obsv. For example, an email sent from Gmail to Yahoo is encrypted in transit via TLS (of the SMTPS protocol [33]) but is accessible in plaintext by both service providers, c.Obsv = {Google, Yahoo}, in addition to the sender and the receiver. We assume the existence of two independent channels c_1 and c_2 between the sender and the receiver, where no adversary controls all communication channels: c_1.Obsv $\cap$ c_2.Obsv = $\emptyset$. While this adversary model is common in theory, ensuring non-collusion in practice where channels may face subpoenas can be challenging. We justify this assumption in §3.

While we use the notion of OTP for simplicity and historical context, our approach is equivalent to n-out-of-n boolean secret sharing, supporting easy generalization to n channels. E2EE communication in EMC^2 works as follows:

EMC^2.Send(M). To send a message M:

(1) The sender uses EMC^2 to secret share the message M by choosing a random key $m_1 \xleftarrow{\$} \{0,1\}^{|M|}$, setting $m_2 \leftarrow M \oplus m_1$, and optionally encoding the shares via Base64.
(2) EMC^2 automatically appends a plaintext preamble explaining how the decryption works at the beginning of m_1 and m_2.
(3) The sender sends the two shares through different channels as c_1.Send(m_1) and c_2.Send(m_2).

$M \leftarrow$ *EMC^2.Receive*(). To receive a message M:

(1) The receiver receives $m_1 \leftarrow c_1$.Recv() and $m_2 \leftarrow c_2$.Recv().
(2) The receiver gets the plaintext preamble explaining the decryption process and how to access EMC^2.
(3) The receiver uses EMC^2 to optionally decode shares and decrypts the message as $M \leftarrow m_1 \oplus m_2$.

Encoding. Some communication channels like email restrict content to printable ASCII characters. Moreover, non-printable characters complicate manual copy&paste of shares between EMC^2 and channels. We include an optional encoding step where the message shares m_1 and m_2 are encoded/decoded using Base64 [24].

2.1 Analysis

Confidentiality. OTP encryption guarantees information-theoretic security as long as the shares (keys) are chosen at random and no adversary can observe all channels. We may use a PRG to generate keys reducing security to a computational guarantee.

Security. The security properties of EMC^2 depend on the properties of the underlying channels. We provide a detailed discussion of the security properties of EMC^2 in §A. However, in favor of simplicity and user understandability, EMC^2 does not offer any guarantee for *integrity, authenticity, or non-repudiation.*

Anonymity. EMC^2 does not offer any *anonymity protection or metadata hiding* mechanisms.

Understandability. EMC^2 consists of three operations: 1) generating a random bitstring, 2) XOR operation, and 3) optional Base64 encoding/decoding. These can be explained through elementary math. To illustrate these concepts, we use coin tossing for randomness generation, bit flipping for XOR, and a look-up table for Base64 encoding. Major programming languages natively support secure randomness generation and Base64 further simplifying our code. Beyond understandability, the simplicity of EMC^2 allows compact implementations that are easy to share, verify, and harden.

Trust. EMC^2 is a fully client-side solution that distributes trust across existing communication channels.

[1] See https://encrypto.de/EMC2 for our demo.

Receiver coverage. EMC^2 requires no setup phase. Senders can initiate E2EE communication via two channels without prior recipient action. This setup-free design separates receiver coverage from the user base, avoiding the cold start problem where limited initial users can hinder adoption. EMC^2 achieves high receiver coverage by bootstrapping on existing channels like email or popular chat apps.

Usability. Requiring users to manually copy&paste shares through two channels adds friction and lower usability. However, mistakes in EMC^2 are low stake, in contrast with PGP accidents like publishing a secret key. As EMC^2 does not rely on underlying channels for E2EE, multi-device support and device recovery can be managed through server-side solutions. We are conducting a user study to assess the usability of EMC^2 as part of future work.

Efficiency. The bottleneck of EMC^2 is the manual sending and receiving of messages. EMC^2 avoids public-key operations and its cost is dominated by its underlying channels. The total communication cost is $O(n|M|)$ bytes, scaling linearly with the number of channels n and the message size $|M|$. While optimizations like using constant-size seeds instead of random shares could reduce the cost to $|M|$, we prioritize the simplicity of the unoptimized approach.

2.2 Automation

The focus of EMC^2 is simplicity and understandability. That is why we decided to leave the process of copy&pasting messages as a manual step. This minimizes the trust surface to a tiny code base and makes the whole process visible to the users. An alternative is automating the whole process for better usability.

Email. Users often use a single application for handling multiple email accounts. Prior work has shown the effectiveness of plugins for automating secret sharing across multiple email providers [8].

Messengers. We have solutions that provide API integration for multiple messengers [6, 52], and new EU regulations such as the Digital Markets Act [18] require interoperability from major messengers. We expect that messaging across different messengers will become possible in the same way that email works across providers. This will facilitate building apps to automate sending OTP encrypted messages across multiple chat channels.

3 Multichannel communication

For over a century, cryptographers assumed that having multiple independent channels between a sender and a receiver is infeasible. However, the proliferation of social media and widespread adoption of point-to-point encryption via protocols like TLS has changed this. Statistical studies on email usage show over 4.2 billion active email users with an average of 1.85 email accounts per person [22, 23]. This 1.85 email addresses per person is attributed to the majority of users having at least two emails to separate accounts for private life, work, and spam. Another study indicates that, on average, users are registered to 6.7 social media and messengers [15]. Many of these platforms surpass one billion active monthly users led by WhatsApp with over 2.4 billion active users [34]. Registration on these platforms often requires a mobile phone number or an email address, which are usually visible and searchable. Nowadays, most Internet communication enables point-to-point encryption (e.g., TLS [38] for messengers and SMTPS [33] for emails), protecting users against eavesdroppers and only revealing (leaking) information to service providers. Given these numbers, it is safe to assume that finding *at least two independent* channels between two users is feasible.

We discuss two common scenarios for secure communication:

Professional. Users often need secure communication with individuals in specific professions like journalists, lawyers, or government officials, where sensitive information is exchanged. In such professional settings, users typically have multiple modes of contact, including private and professional email addresses and mobile phone numbers, linked to various messengers. Additionally, professionals often maintain accounts on social media platforms like LinkedIn and X (formerly Twitter), enabling users to find multiple contact channels without directly requesting them.

Social. Sending sensitive social messages often happens with people personally known by the sender. In such scenarios, it is common to know the recipient's private email addresses, mobile phone numbers and connected messengers, and social media accounts on platforms like Facebook, Instagram, TikTok, or Snapchat. These social platforms also provide messaging services.

We assume no adversary can control or observe all channels as mentioned in §2. Since EMC^2 allows users freedom in channel selection, we can't enforce this directly. Users should be aware of three scenarios that break this assumption: 1) If an adversary compromises a user's device, they can observe all communications. 2) When an entity controls multiple platforms (e.g. WhatsApp, Facebook Messenger, and Instagram by Meta), it observes multiple channels that may look independent. Thus, users should use channels from different companies. 3) Subpoenas can compel service providers in a single jurisdiction to reveal data, so channels from different jurisdictions are recommended. While high-resource adversaries may still perform targeted attacks, mass surveillance of EMC^2 users and requiring monitoring from messengers would be cost-prohibitive.

4 Related work

We systematize existing tools and approaches to achieve E2EE communication and how they interact or compare with EMC^2.

4.1 Email

PGP. Encrypted email protocols like PGP and S/MIME are one of the first attempts at achieving E2EE. Unfortunately, they are notorious for their complex setup process and severe usability issues [40, 56]. The general audience struggles to understand the inner workings and properties of PGP. Additionally, mistakes, such as sharing the secret key instead of the public key, can be catastrophic. PGP has a low adaptation rate compared to email use which further limits its functionality as sending a secure message before the receiver generates and shares their key is not possible.

Autocrypt. The primary challenge of PGP is securing the secret key and distributing the public key. There are community efforts aiming to make PGP key management transparent, such as the AutoCrypt protocol [4], supported by mail agents like DeltaChat [16] and K9 [25]. AutoCrypt-enabled mail agents automatically create public keys and propagate them through SMTP headers without interfering with generic email providers. This enables opportunistic encryption when the receiver's public key is known, using PGP and S/MIME directly. However, AutoCrypt still faces low adoption and key management issues in multi-device scenarios.

Secure email providers. Some email providers are security-focused like Proton [36], StartMail [44], and Tuta [51] and support E2EE. These solutions automatically generate and manage PGP keys for users, where secret keys are encrypted with a password only known to the client and stored on the server. These services use PGP encryption when the receiver belongs to the same provider or has a known public key. If there is no known receiver public key, the mail is encrypted via a symmetric key. The sender has to send this symmetric key to the receiver via an out-of-band channel and the provider offers a web portal to assist the decryption.

Prior work by Kobeissi [26], has studied the E2EE properties of these secure email providers and raised concerns about 1) the use of low-entropy passwords to protect secret keys from providers and 2) reliance on large cryptographic code bases served as dynamic websites by providers to protect users' secret messages and passwords. In contrast, EMC^2 requires no password or long-term keys, and the code is minimal and runs offline.

PrivMail. The closest work to EMC^2 is PrivMail [8], which secret shares emails across multiple email providers. PrivMail provides keyword search over secret shared emails to facilitate server-side monitoring and spam filtering. EMC^2 supports arbitrary communication channels and focuses on understandability and verifiability.

4.2 Messengers

The most popular way to achieve E2EE is using secure messengers such as Signal [43], Telegram [46], Threema [47], and WhatsApp [55]. Regulations such as the Digital Market Act [18] demand interoperability between major messengers. The main contenders for an interoperable protocol are Matrix [31], MLS [5], and Signal [1], all providing E2EE. At first glance, this may solve the problem of E2EE communication, but concerns remain. Many large messengers, like WhatsApp with over 2 billion active users, are proprietary software where users cannot see or verify what protocol is running and have to trust the service provider blindly. Open-source alternatives like Signal, despite high security perception, have smaller user bases (40 million active Signal users) limiting their coverage and functionality compared to email or larger messengers. Moreover, all above mentioned messengers rely on central service providers. While they offer theoretical guarantees for E2EE, their security mechanism is not visible or understandable by the general public and they are susceptible to regulations mandating backdoors.

EMC^2 can achieve high receiver coverage by allowing senders to initiate secure messaging without requiring receivers to install an app. Running EMC^2 over E2EE messengers is an excellent solution to distribute trust across multiple service providers. By providing a simple and understandable client-side E2EE solution, EMC^2 demonstrates the ineffectiveness of mandating backdoors.

4.3 Anonymous communication systems

Stand alone. Anonymous communication systems, often providing E2EE, have been thoroughly studied over the years. Various approaches include: Crowds [37], DC-net [10, 11, 21], Differential privacy [28, 29, 54], Mixnet [9, 13, 27], PIR [2, 3], and traffic-shaping [17, 35]. However, these systems are bound by the anonymity trilemma [14] stating that each solution can only achieve two properties out of strong anonymity, low bandwidth overhead, and low latency. This often leads to higher costs or delays. Supporting metadata protection is crucial for many sensitive scenarios, but to the best of our knowledge, none of these solutions have achieved a large enough user base for our receiver coverage goal.

While EMC^2 can use anonymous communication systems as the underlying channels, the privacy of the system will be limited to the weakest channel metadata protection offered at best. Therefore, EMC^2 is not suitable for use cases requiring strong anonymity.

Tor hidden services. Anonymous messengers like Briar [7], Cwtch [12], Ricochet [39], and Tox [48] build peer-to-peer (P2P) messaging using Tor hidden services. These solutions require both the sender and receiver to be online simultaneously and do not support asynchronous communication. These messengers are often used in niche scenarios, have a limited user base (millions), and receiver coverage is restricted to those who have installed the app.

Web3. There are efforts to build decentralized web3 anonymous messengers that operate on top of a decentralized network or a blockchain such as Session [42], Status [45], and XX messenger [32]. While decentralization helps with removing the central trust, these solutions suffer from a small user base and receiver coverage too.

4.4 Embedded messengers

Many organizations handling sensitive information, such as banks, legal firms, and journalist groups, provide mobile apps offering online functionality and support. Many apps allow users to message the organization's employees. These apps often hardcode the organization's public keys or rely on TLS encryption, but as they have a first-party server that is the intended recipient of the message, they arguably provide E2EE. While these apps enable secure communication for some sensitive roles like lawyers and journalists, creating, deploying, and verifying a secure messaging app for every organization needing confidential contact is not a scalable solution.

5 Discussion and conclusion

We designed EMC^2, an E2EE solution that uses n-out-of-n secret sharing to distribute trust across n communication channels providing point-to-point encryption. EMC^2 can be used over arbitrary existing and widely deployed channels and does not require any setup; this allows supporting a high receiver coverage. The sender needs to know about EMC^2. However, the receiver just follows a short explanation embedded in the secret shares to read encrypted messages, without having to install an application or register for a service.

Simplicity. Many features of EMC^2 are enabled by its simplicity. Simplicity is the cornerstone of our understandability by a general audience and enables building compact solutions with a few lines of code that are easy to share, harden, verify, and audit.

Use cases. The primary design goal of EMC^2 is educating the public about E2EE and demonstrating the ineffectiveness of legal mandates for backdoors. EMC^2 complements existing E2EE solutions and does not aim to replace them. EMC^2 is effective in cases where conversation confidentiality is the main goal such as contacting lawyers and sending medical or financial data. However, EMC^2 is not suitable for all scenarios including cases involving journalists or activists where protecting metadata is crucial.

Future work. We are currently running a user study to investigate the usability of manually copy&pasting messages and to

measure the impact of understandability on trust. Moreover, it is possible to extend EMC^2 to support E2EE group messaging.

Acknowledgements. This project was funded by the DFG within SFB 1119 CROSSING/236615297 and GRK 2050 Privacy & Trust/251805230. It also received funding from the ERC under the European Union's Horizon 2020 research and innovation program (grant agreement No. 850990 PSOTI). ChatGPT was used for text refinement to improve clarity.

References

[1] Joël Alwen, Sandro Coretti, and Yevgeniy Dodis. The double ratchet: Security notions, proofs, and modularization for the signal protocol. In *Eurocrypt*, 2019.
[2] Sebastian Angel, Hao Chen, Kim Laine, and Srinath T. V. Setty. PIR with compressed queries and amortized query processing. In *IEEE S&P*, 2018.
[3] Sebastian Angel and Srinath T. V. Setty. Unobservable communication over fully untrusted infrastructure. In *USENIX OSDI*, 2016.
[4] AutoCrypt Project. Autocrypt: E-mail encryption settings discovery. https://autocrypt.org.
[5] Richard Barnes, Raphael Robert, and Joe Hildebrand. The Messaging Layer Security (MLS) Protocol. https://datatracker.ietf.org/doc/rfc9420, 2023.
[6] Bird. Bird - omnichannel communication solutions. https://www.bird.com.
[7] Briarproject. Briar: Secure messaging anywhere. https://www.briarproject.org.
[8] Gowri R. Chandran, Raine Nieminen, Thomas Schneider, and Ajith Suresh. Privmail: A privacy-preserving framework for secure emails. In *ESORICS*, 2023.
[9] David Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM*, 1981.
[10] Henry Corrigan-Gibbs, Dan Boneh, and David Mazières. Riposte: An anonymous messaging system handling millions of users. In *IEEE S&P*, 2015.
[11] Henry Corrigan-Gibbs and Bryan Ford. Dissent: accountable anonymous group messaging. In *ACM CCS*, 2010.
[12] Cwtch. Surveillance Resistant Infrastructure. https://cwtch.im.
[13] George Danezis, Roger Dingledine, and Nick Mathewson. Mixminion: Design of a type III anonymous remailer protocol. In *IEEE S&P*, 2003.
[14] Debajyoti Das, Sebastian Meiser, Esfandiar Mohammadi, and Aniket Kate. Anonymity trilemma: Strong anonymity, low bandwidth overhead, low latency - choose two. In *IEEE S&P*, 2018.
[15] Datareportal. Global social media statistics. https://datareportal.com/social-media-users.
[16] Delta Chat. Delta chat is a decentralized and secure messenger. https://delta.chat.
[17] Kasra Edalatnejad, Wouter Lueks, Julien Pierre Martin, Soline Ledésert, Anne L'Hôte, Bruno Thomas, Laurent Girod, and Carmela Troncoso. Datasharenetwork: A decentralized privacy-preserving search engine for investigative journalists. In *USENIX Security*, 2020.
[18] European Commission. The digital markets act. https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX%3A52020PC0842, 2020.
[19] European Commission. Proposal for a Regulation of the European Parliament and of the Council laying down rules to prevent and combat child sexual abuse. https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=COM:2022:209:FIN, 2022.
[20] Simson L. Garfinkel. *PGP - pretty good privacy: encryption for everyone.* O'Reilly, 1995.
[21] Sharad Goel, Mark Robson, Milo Polte, and Emin Sirer. Herbivore: A scalable and efficient protocol for anonymous communication. Technical report, Cornell University, 2003. http://www.cs.cornell.edu/people/egs/papers/herbivore-tr.pdf.
[22] Radicati Group. Email Statistics Report, 2018-2022. https://www.radicati.com/wp/wp-content/uploads/2018/01/Email_Statistics_Report,_2018-2022_Executive_Summary.pdf.
[23] Radicati Group. Email Statistics Report, 2023-2027. https://www.radicati.com/wp/wp-content/uploads/2023/04/Email-Statistics-Report-2023-2027-Executive-Summary.pdf.
[24] S. Josefsson. The base16, base32, and base64 data encodings. https://datatracker.ietf.org/doc/rfc4648, 2006.
[25] K9 mail. Advanced email for android. https://k9mail.app.
[26] Nadim Kobeissi. An analysis of the protonmail cryptographic architecture. *IACR Cryptol. ePrint Arch.*, 2018. Paper 2018/1121. https://eprint.iacr.org/2018/1121.
[27] Albert Kwon, David Lazar, Srinivas Devadas, and Bryan Ford. Riffle: An efficient communication system with strong anonymity. *PoPETs*, 2016.
[28] David Lazar, Yossi Gilad, and Nickolai Zeldovich. Karaoke: Distributed private messaging immune to passive traffic analysis. In *USENIX OSDI*, 2018.
[29] David Lazar, Yossi Gilad, and Nickolai Zeldovich. Yodel: strong metadata security for voice calls. In *ACM Symposium on Operating Systems Principles (SOSP)*, 2019.
[30] Hyunwoo Lee, Doowon Kim, and Yonghwi Kwon. TLS 1.3 in practice: How TLS 1.3 contributes to the internet. In *The Web Conference (WWW)*, 2021.
[31] Matrix. Matrix: An open network for secure, decentralized communication. https://matrix.org.
[32] XX messanger. XX messenger: Building a world where your life belongs to you. https://xx.network.
[33] K. Moore and C. Newman. Cleartext Considered Obsolete: Use of Transport Layer Security (TLS) for Email Submission and Access. https://datatracker.ietf.org/doc/rfc8314, 2018.
[34] Bussiness of apps. App statistics: Messaging apps. https://www.businessofapps.com.
[35] Ania M. Piotrowska, Jamie Hayes, Tariq Elahi, Sebastian Meiser, and George Danezis. The loopix anonymity system. In *USENIX Security*, 2017.
[36] Proton. Proton Mail: Secure email that protects your privacy. https://proton.me.
[37] Michael K. Reiter and Aviel D. Rubin. Crowds: Anonymity for web transactions. *ACM TISSEC*, 1998.
[38] Eric Rescorla. The Transport Layer Security (TLS) Protocol Version 1.3. https://datatracker.ietf.org/doc/rfc8446, 2018.
[39] Ricochet. Ricochet Refresh: an open-source project to allow private and anonymous instant messaging. https://www.ricochetrefresh.net.
[40] Scott Ruoti, Jeff Andersen, Daniel Zappala, and Kent E. Seamons. Why Johnny Still, Still Can't Encrypt: Evaluating the Usability of a Modern PGP Client. *CoRR*, abs/1510.08555, 2015. https://arxiv.org/abs/1510.08555.
[41] Jim Schaad. Secure/multipurpose internet mail extensions (s/mime) version 4.0 message specification. https://datatracker.ietf.org/doc/rfc8551, 2019.
[42] Session. Session messenger: Send encrypted messages, not metadata. https://getsession.org.
[43] Signal. Signal messenger. https://signal.org.
[44] StartMail. StartMail: Secure email that puts your privacy first. https://www.startmail.com.
[45] Status. Status messenger: Chat privately with friends. https://status.app/features/messenger.
[46] Telegram. Telegram: a new era of messaging. https://telegram.org.
[47] Threema. Threema: The secure messenger for individuals and companies. https://threema.ch.
[48] Tox. Tox: A New Kind of Instant Messaging. https://tox.chat.
[49] Perrin Trevor and Marlinspike Moxie. The double ratchet algorithm. *GitHub wiki*, 2016.
[50] Carmela Troncoso, Bart Preneel, et al. An open letter to the european commission: Concerns over new proposed child sexual abuse regulation. https://cdt.org/wp-content/uploads/2023/05/2023-05-16-Letter-from-Public-Interest-Technologists.pdf, 2023.
[51] Tuta. Tuta: Secure email made for you. https://tuta.com/security.
[52] Twilio. Twilio - Communication APIs for SMS, Voice, Video and Authentication. https://www.twilio.com.
[53] United States Senate. S.1207 - earn it act of 2023, 118th congress. https://www.congress.gov/bill/118th-congress/senate-bill/1207, 2023.
[54] Jelle van den Hooff, David Lazar, Matei Zaharia, and Nickolai Zeldovich. Vuvuzela: scalable private messaging resistant to traffic analysis. In *ACM Symposium on Operating Systems Principles (SOSP)*, 2015.
[55] WhatsApp Inc. Whatsapp messenger: secure and reliable free private messaging and calling. https://www.whatsapp.com.
[56] Alma Whitten and J. Doug Tygar. Why Johnny Can't Encrypt: A Usability Evaluation of PGP 5.0. In *USENIX Security*, 1999.

A Impact of underlying channels on security

In EMC^2, the sender can freely choose the underlying channels. The security properties of these channels greatly affect the overall security of the communication system. Users often lack full knowledge of these channel properties, and presenting them with a detailed channel-based analysis would conflict with our goal of simplicity and understandability. Therefore, we do not make claims to guarantee *integrity, authenticity, non-repudiation, and deniability*. Though, we provide an informal analysis for interested readers.

The classic OTP cipher sends message shares through insecure channels, assuming the adversary cannot observe all share. This method offers no integrity or authenticity protection, but users can achieve deniability easily. Given a communication transcript $M = m_1 \oplus m_2$ and a desirable fake message M', users can forge a share $m_1' = M \oplus M'$ to decrypt m_2 into the fake message M'.

EMC^2 supports various mediums, such as email and WhatsApp, each offering different security properties. For instance, we assume all channels are protected by a series of encrypted point-to-point

links using protocols like TLS. This transport layer protection ensures that network adversaries cannot break the integrity of channels, and the attack surface is limited to channels' service providers, c.Obsv. Now we discuss security properties in more detail.

Integrity. The integrity of communication in EMC^2 depends on the weakest integrity protection among the underlying channels. Any modification to a message requires altering at least one of the message shares. If all channels protect the integrity of message shares m_i, we can infer that the combined message $M = \bigoplus m_i$ remains unmodified.

Authenticity. Like integrity, the authenticity of EMC^2 relies on the weakest authenticity protection among the channels.

Deniability. Communication remains deniable if one of the underlying channels is deniable or lacks integrity protection. As long as a user can alter or deny one of the shares undetected, the deniability proof of the classic OTP cipher holds.