Peace vs. Privacy: Leveraging Conflicting Jurisdictions for Email Security

Mohammad Mannan, Arash Shahkar, Atieh Saberi Pirouz and Vladimir Rabotka
Concordia Institute for Information Systems Engineering
Concordia University, Montreal, Canada
m.mannan@concordia.ca

ABSTRACT

We introduce the paradigm of *security through hostility*: cloud-based service providers in conflicting jurisdictions are assumed to be non-cooperating, and are used for transmitting encrypted content and corresponding keys through separate but accessible channels among end-users from both jurisdictions. Such separation between content and key can enable effortless user-to-user encrypted communication without any user-managed keys. As an example use-case of this paradigm, we consider encrypted email, which is complicated by the requirement of balancing security and ease-of-use needs. For example, users cannot be expected to manage long-term keys (e.g., PGP key-pair), or understand crypto primitives. We design *CherAmi* by leveraging existing relationships between a sender and a receiver on an online social networking (OSN) site, and assuming users can use OSN and email providers that are hosted from hostile/non-cooperating jurisdictions. CherAmi can provide integrity, authentication and confidentiality guarantees for selected messages among OSN friends. A confidentiality-protected email is encrypted by a randomly-generated key, and the key is *privately* shared with the receiver via the OSN site. Our implementation consists of a Thunderbird add-on and a Twitter app; the add-on is available at: https://madiba.encs.concordia.ca/software.html. CherAmi is a client-end solution and does not require changes to email or OSN servers. In this paper, the focus of our discussion includes: the paradigm of security through hostility, and the design, implementation and security analysis of the proposed encrypted email solution. We acknowledge that a user study will be required to validate usability-related features of CherAmi.

CCS Concepts

•**Security and privacy** → **Key management; Authentication; Social aspects of security and privacy;** *Access control; Social network security and privacy;*

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

NSPW '15, September 08-11, 2015, Twente, Netherlands
© 2015 ACM. ISBN 978-1-4503-3754-0/15/09…$15.00
DOI: http://dx.doi.org/10.1145/2841113.2841123

Keywords

Surveillance, Email encryption, Key distribution, State-level hostility

1. INTRODUCTION

Wars and conflicts are an integral part of the human history, as exemplified by the war between Sumer and Elam in Mesopotamia in 2700 BC to recent incidents between USA-"Axis of Evil", Russia-Ukraine, China-Japan, and China-USA (for references to ancient wars, see, e.g., [25, 32]). As long as conflicts are inevitable, we explore how they can be leveraged for online security. Internet has enabled the so called cloud-based services that are hosted across geographical boundaries, and appear mostly transparent to end users. Although several large cloud services are run by US companies (e.g., Google, Facebook, Dropbox), which attracted a global audience, many small-scale regional/nationalistic services exist (e.g., Chinese QQ, Russian VK); efforts to build more local services may be even increasing due to recent Snowden revelations [52].[1]

Even though the use of user-level encryption could improve privacy against adversaries including the NSA, such encryption is rarely used (see e.g., [59]). An apparent reason is the complexity of key management, especially when public key cryptography is used.[2] By hosting encrypted content and the corresponding (symmetric) keys in non-cooperating jurisdictions, we propose to avoid user-level key management issues. Note that our approach is different than (k, n) threshold schemes [45, 4], where an adversary is assumed to have at most access to $k-1$ pieces (out of a total of n), but not k, which is the minimum number of pieces required to recover the secret. We also do not rely on sending multiple pieces of a secret via different channels, assuming an adversary cannot control all such channels (cf. [37, 8]). Instead, we use existing user-to-host SSL/TLS channels (assumed to be secure) to transfer the key and the ciphertext through different services operating in *hostile* jurisdictions. We require well-known hostile entities, not merely independent parties under the same legal jurisdiction, unlike the defunct Clipper

[1]For example, see remarks made by the former NSA deputy director of training Cedric Leighton [64]: "When you have a situation where all of a sudden, everyone goes into 'tribal' mode – a German cloud, a Swiss cloud, or any other separate internet, they are significant nationalistic attempts." See also the recent Russian law about *data residency* [51].
[2]Cf. Bruce Schneier (preface of [44]): "Key management is the hardest part of cryptography and often the Achilles' heel of an otherwise secure system."

Chip proposal, where a escrow key was split into two parts, and each part was then shared with distinct US government agencies or private custodians (cf. [16, 5]). Our approach can be viewed as a (2, 2) threshold scheme, where at most one of the key and ciphertext is available to an adversary, due to our particular choice of the third-party services.

The primary goal of our approach is to make key management transparent to users, and thereby, increase adoption of end-to-end encrypted communication to hinder mass surveillance programs. Obviously, military or enterprise-grade security is a non-goal (cf. [56]). If conflicting regions start to cooperate, our approach loses, but that is not so unsavory of an outcome—an *optimistic* view is that peace may win, resulting in less human suffering (cf. [58]). More traditional approaches (e.g., S/MIME, PGP) may be used when such peaceful scenarios, if ever, come to play.

As a concrete use-case of the above-mentioned security via hostility paradigm, we explore the current sorry state of email security. Billions of emails are exchanged everyday, with almost all of them being stored/available in plaintext to third parties. Users are seemingly finding it acceptable that a few email providers have complete access to their most intimate messages. This incredible paradigm shift took place within the span of only a few decades. On the legal side, as revealed by the operator of currently defunct encrypted email provider Lavabit, what rights users have are apparently questionable even in the most powerful democracy in the world: "...the prosecution also argued that my users had no expectation of privacy..." [55]. Seemingly, the current situation is the result of several factors, including: most people are unaware that their emails are not private at all, which may be attributed to the email/Internet infrastructure being transparent to average users (cf. [24]); a wide-spread misconception among users is "I've got nothing to hide" [47]; and the inadequacy of existing email security solutions (e.g., PGP, S/MIME, STEED [26], and Aquinas [8]), as analyzed in several studies [61, 19, 38].

We propose *CherAmi*,[3] a secure email technique designed for everyday users as well as online activists, who are also connected via an online social networking (OSN) service, using Russia-based mail.ru, China-based QQ or other similar services as the email provider and US-based Twitter as the OSN service. Cyber conflicts between Russia, China and the USA are on the rise (e.g., [6, 49]), and we assume these countries remain unfriendly enough not to share their surveillance data in the near future. In CherAmi, an email client add-on creates a per-message symmetric key to encrypt the email content; cf. the use of symmetric key based message encryption in privacy-enhanced electronic mail standards such as RFC 1421. The key is then published securely (e.g., via an SSL-protected channel) on the OSN site, which is instantly accessible to the receiver, e.g., as a direct private message in Twitter. A similar add-on in the recipient's email client accesses the OSN on behalf of the user to retrieve the message key; the add-on verifies the ciphertext and decrypts the email content. The sender's authenticity is verified implicitly by the OSN site, as the key value is accessible only to the receiver through the pre-existing social relationship. Confidentiality is maintained by the per-message encryption key. Although the process may appear sophisticated, CherAmi's implementation provides the user with a transparent experience, and requires only modest user involvement (mostly during the initial configuration). See Fig. 1 for a brief overview.

[3]We choose this name from the World War I carrier pigeon named "Cher Ami" (meaning, "Dear Friend"), which flew over enemy lands and delivered messages that saved hundreds of lives; see http://www.si.edu/Encyclopedia_SI/nmah/cherami.htm.

We use existing OSN sites for key transport, to simplify the key sharing and verification process, which has been identified as an important barrier to PGP's adoption [61] (see also [42]). Note that, the use of OSN sites for sharing long-term public keys is not new; see e.g., Waterhouse [28] (more details in Section 7). However, we leverage popular OSN sites as an instantly-accessible, authenticated channel for distributing per-email symmetric keys, largely avoiding the key distribution and key management issues of existing public-key based solutions.

In contrast to other solutions that target *all* email users, but are scarcely adopted in practice (if at all), we limit CherAmi to enable secure emails primarily among *friends*, more specifically, Twitter contacts who supposedly *follow* each other. Our hope is that CherAmi may be more easily adopted due to this targeted user base and transparent key management. CherAmi is designed considering Twitter as the OSN service and implemented as a Mozilla Thunderbird add-on. Thus, users have the freedom to use their email provider of choice (as long as the provider operates from a country hostile towards the US), making the solution available to a large number of users. There are about 288 million monthly active Twitter users as of April 2015.[4] CherAmi can be extended to be used with other OSN services too.

In this paper, we report primarily on the design, implementation and security analysis of CherAmi, as an example solution enabled by the security through hostility paradigm. We discuss challenges in implementing the seemingly straightforward design into existing popular services.[5] The discussion of these challenges may help future efforts in designing and implementing practical secure email solutions. While a key step in validating CherAmi's adoption incentives and usable key management, a comprehensive user study is being considered as separate future work.

Contributions.

1. New paradigm for cyber-security. The existing security through hostility paradigm has used hostility to support matters such as personal safety, and increasing the difficulty of tracing attacks. It has not been available to ordinary, everyday users. We adapt that paradigm to ordinary users to increase their privacy, by storing encrypted content and the corresponding key in mutually hostile jurisdictions. The new paradigm can address key management barriers in cloud-based services for end-to-end encryption. As a concrete example, using this paradigm, we design a secure email system called CherAmi; the paradigm can also be easily adapted for other cloud services.
2. Transparent key exchange. Unlike most other so-

[4]https://about.twitter.com/company

[5]Parts of CherAmi are related to an unpublished but publicly available past mechanism (called *FriendlyMail* [40]), relying on competing businesses that are expected to be non-cooperating. This past tool used Facebook as the OSN to secure Gmail messages. However, after revelations of the Prism program [53], the usual non-cooperation among businesses for not sharing user data cannot be relied on.

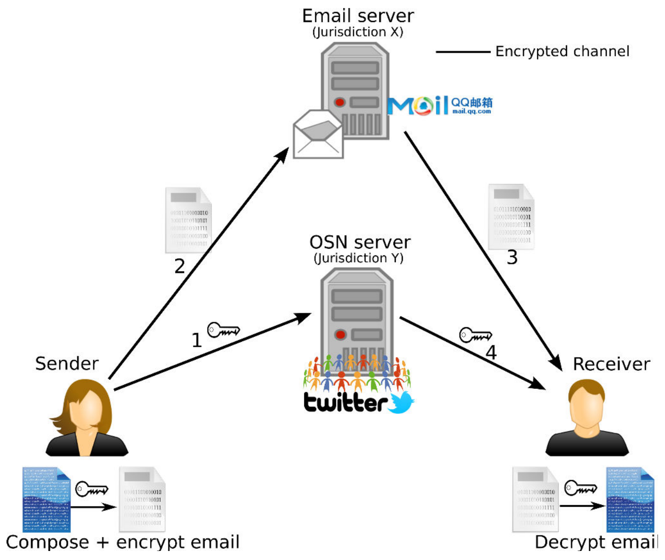


Figure 1: Simplified CherAmi steps: (1) a per-message randomly generated key is shared with the recipient via an OSN site; (2) the encrypted email message is sent via an email provider; (3) the recipient receives the encrypted email; and (4) the email content is decrypted by the per-message key retrieved from the OSN. Jurisdictions X and Y are assumed to be non-cooperating.

lutions, CherAmi does not require the receiver of a confidential email to create keying materials (as in public key systems) *before* she can receive such emails. The only pre-requisite for two individuals to exchange confidential email messages is to be following each other on Twitter.

3. No server-side changes. No changes to the server-side of the social networking sites or email providers are needed. OSN-connected users can immediately benefit from CherAmi.
4. Gradual adoption. CherAmi allows a sender to indicate which emails should have integrity and confidentiality protections. Default encryption of all emails would be more privacy-preserving; however, such a design may not work for many users (due to e.g., not using OSNs).
5. Implementation. To evaluate the feasibility of our design, we have implemented CherAmi as a Thunderbird add-on and a Twitter app. Basic email features, e.g., message compose, send, receive, reply, forward, attachment-support, and multi-party email have been implemented. We discuss several implementation challenges that highlight complications of implementing privacy-protection mechanisms on top of popular real-world services.

2. THREAT MODEL AND OPERATIONAL ASSUMPTIONS

The threat model and operational assumptions discussed here are mostly applicable to our email security proposal (CherAmi). Some of these assumptions can be generalized for applications relying on the security through hostility paradigm. However, the threat model should be reconsidered when designing new applications or services based on this paradigm.

Hostility/non-collusion assumptions between jurisdictions. The fundamental assumption for the security through hostility paradigm is that the jurisdictions hosting the target services will not combine data from both services. For CherAmi, the email and OSN providers must be separate, non-colluding entities, residing in countries hostile to each other (e.g., between US and Russia or China). At the minimum, we require providers to be located in different legal jurisdictions that generally do not share user data (e.g., between US and EU countries). Either of the service providers may cooperate with an adversary, providing the adversaries do not work together. *CherAmi is only as secure as the level of non-cooperation between the providers.* If competing providers from the same jurisdiction (e.g., US-based Twitter and Gmail) are used, users will be at the risk of being exposed to government agencies of that jurisdiction (cf. the Prism surveillance program [53]); thus, we do not recommend such providers. On the other hand, using services hosted from hostile jurisdictions may pose usability and deployability challenges; e.g., Twitter is officially blocked in China as of October 2015. Note that, determined users can still access such services via anti-censorship tools including Tor.

Technical sophistication of enemy states. Differences in technical sophistication and political power between hostile jurisdictions may also need to be considered; i.e., not all enemy state pairs are the same such as US/China vs. US/Venezuela. It might be easier for US agencies to compromise a service from Venezuela than China (especially, services backed by the Chinese government). Thus, in addition to mutual hostility, the interested parties must not be able to subvert each other's infrastructure.

Providers under multiple jurisdictions. Some large email/OSN providers are multi-national—i.e., must abide by local laws in multiple jurisdiction; e.g., Google is accountable to both the US and European Union. To safely use Gmail, users must choose an OSN provider hostile towards both jurisdictions. Also, for cloud storage/application services, defining legal boundaries may be tricky; see e.g., Hoboken et al. [23], for how US laws (Patriot Act/FISA) can be used to access user data in EU countries. The ongoing (as of October 2015) Microsoft Ireland vs. US case may clarify the reach of cross-border data access laws; see e.g., [57].

Hidden cooperation between enemy states. Any mechanism (such as CherAmi) relying on the security through hostility paradigm is vulnerable to publicly unknown collaboration between seemingly hostile nations. Enemy states, specifically their intelligence agencies and critical business entities occasionally collaborate, sometimes out of necessity, as apparent from many past incidents; examples include: Iran-West cooperation on narcotics [33], China-Russia [39], and partnerships between intelligence agencies of otherwise unfriendly nations [3]. Such hidden cooperation is detrimental to privacy protections as enabled by our approach, but may at the end promote peace between nations, and thus reduce human suffering. Also, as a direct result of an Internet-connected world, backdoor collaboration may be more easily exposed and widely reported through social media. When such collaboration is revealed, or when old enemies become friends (e.g., US and Japan after World War II), users must switch services (e.g., replace Mail.ru with QQ mail), or adopt more traditional approaches (e.g., S/MIME, PGP). Information sharing agreements such as mutual legal assistance treaties (MLATs) between hostile jurisdictions (if one exists) may also need to be considered.[6]

Channel and host security. Network connections between users and OSN/email servers must be protected (e.g., via SSL). The user-to-OSN channels must be secure for obvious reasons (i.e., to protect key values). If user-to-email service channels are not encrypted, the OSN site may break email confidentiality if it can collect the encrypted content. Note that, Internet traffic sometimes crosses international boundaries even for communications between users within the same country (cf. *boomerang routing* [36]). The use of mobile broadband for the email content may alleviate such concerns (i.e., routed via local mobile infrastructures). A mobile operator can also be used as an OSN, when a foreign email service is used; e.g., for US users, Mail.ru can be used with keys sent as SMSes via Verizon Wireless.

We also assume a Dolev-Yao [13, 35] network adversary, with no control over end-user machines. Both the sender and receiver-end machines are assumed to be malware-free; otherwise, malware can simply expose or modify the email content while being composed or read. We exclude recently revealed clandestine government-sponsored efforts to weaken widely-used crypto systems that are building blocks of a secure channel (see e.g., [21, 54]); attacks against the SSL CA infrastructure (see e.g., [46]); and attacks that are directed towards compromising end-user machines (see e.g., [43]). On the positive side, such attacks and backdooring/exceptional access efforts generally do not remain hidden for long and their efficacy is also questionable (see e.g., [52, 46, 1]).

Assumptions on OSN and email providers. With regards to the email content, we assume that the email and OSN providers are non-malicious but curious entities; i.e., they will provide their services in the usual manner, but would prefer to learn the email content, if possible (e.g., for advertisements, building elaborate user profiles). We use the OSN provider for sender origin authentication, and assume that OSN profiles and connections between users are largely genuine. OSN providers, e.g., Facebook and Twitter actively try to discover and remove fake profiles. However, such profiles are still a significant concern (see e.g., [7]). Detecting fake and compromised accounts is also an active research area (e.g., [9, 15]). We also require that the OSN providers can protect the confidentiality and integrity of user posts, e.g., to not expose privately posted keys to unauthorized parties. Similarly, we assume email providers can protect user emails from unauthorized parties (but see e.g., [22]). At least, for the sake of their business reputation, email/OSN providers are apparently diligent in fixing known vulnerabilities and maintaining their site security.[7] Several cloud services including Google, and Twitter, are also now encrypting communications within their data centers; for a list of companies that encrypt their data center links, see EFF (eff.org/encrypt-the-web-report).

OSN trust relationships. We assume that the OSN trust-relationship between a sender and receiver (i.e., OSN connections or lack thereof) can be determined from their email applications. For example, the receiver's email address or full-name can be searched in the sender's OSN contact list to verify if they have a direct connection. Currently, several OSN providers, including Twitter, Facebook, QQ, provide API support to programmatically access a user's OSN contact list from an email add-on or other applications.

Key deletion and other assumptions. To prevent perpetual access to an email's content, users can delete keys from the OSN site after an encrypted message has been retrieved, or after a given time period. Such key deletion will help keep past messages confidential even when enemy states start to cooperate. However, OSN sites may not actually delete any posts for a long period of time; see, e.g., Facebook policy on deleted content.[8] Thus, immediate message *self-destruction* (e.g., Vanish [20]) is a non-goal (which is rather difficult to achieve, cf. [62]).

As a sender, the adversary may impersonate a friend, or a known company (e.g., the user's bank); i.e., the `From` field can be arbitrary, and we do not assume any other sender verification techniques being used (e.g., SPF/DKIM). A user's email and OSN account credentials must not be compromised; we discuss consequences of such compromises in Section 5. Also, the recipient of a confidential email is trusted not to share/forward the content with unauthorized parties. Privacy of communications metadata (e.g., email header, including sender and receiver email addresses, and subject lines) is also not provided; see the MIT Immersion Project[9] as an example of how revealing email metadata can be.

3. USER STEPS AND VARIANTS

In this section, we detail CherAmi and user steps for sending/receiving confidential and integrity-protected emails; see Table 1 for notation used. We describe the steps necessary for Alice to send a protected email to Bob. This scheme will be applicable to multiple recipients as well. Parts of these steps are explained through our prototype for Thunderbird (as an add-on) and Twitter (detailed in Section 4).

For the primary mode of operation, CherAmi assumes a direct OSN trust relationship between users. In Section 3.2, we consider other trust relationships, and also discuss several variants to our proposal.

Design overview. CherAmi leverages symmetric key cryptography and OSN integration, to enable confidentiality for email content and integrity verification for both content and header information, including origin, subject and list of recipients. The overall design is related to the well-established notion of using multiple channels to achieve security goals (e.g., [63, 31]), but with a crucial difference: we want stored encrypted content and keys to be hosted by services from conflicting countries. CherAmi employs OSN sites as an additional channel, mainly to automate key management. Secure emails are communicated using email providers as the main channel. Existing pre-authenticated connections

[6]For example, see the US Department of State (http://www.state.gov/j/inl/rls/nrcrpt/2014/vol2/222469.htm) for a list of countries that signed such agreements with the US government.

[7]In comparison, PGP public key servers also do not always function as intended; known issues include: older version usage, key synchronization problems (see e.g., [11]).

[8]http://www.facebook.com/help/356107851084108/

[9]https://immersion.media.mit.edu

ID_A, ID_B	Unique email addresses for Alice (sender) and Bob (receiver) respectively.
$Subject$	Subject line of an email message composed by Alice.
M	Content of an email composed by Alice, excluding header information.
K_m, K_h	Per-email, randomly generated symmetric keys of adequate length (e.g., 128 bits) for message encryption and message authentication code (MAC) computation, respectively.
$E_{K_m}(\cdot)$	An authenticated, symmetric-key based encryption function (e.g., AES in the CCM mode) with key K_m.
$MAC_{K_h}(\cdot)$	A message authentication code (MAC) function with key K_h.
C_m	Content of an email message encrypted by Alice (email body only, excluding email headers).
C_h	MAC of an email header information.
C	Content of a protected email sent via CherAmi, excluding header information.
M_{id}	Unique identifier for a private message sent from Alice to Bob on the OSN site. The OSN site is assumed to prevent unauthorized access to private messages even if this identifier is known; see Section 2.
$Hdr(\cdot)$	A dynamic but distinguishable header for C.
Ftr	A fixed footer appended to C.
$x\|\|y$	x concatenated with y.
Mrk_k, Mrk_m	Separator markers for keys and message parts respectively.

Table 1: Notation used in CherAmi

among OSN users are leveraged to exchange secrets between email senders and receivers.

3.1 Protected Emails

Sending an encrypted email. We require explicit user selection for protected messages. Users may indicate their desire to send a protected message by turning on a *Confidential* check-box, added to the regular email compose window interface. To emphasize that a protected message is to be sent, turning on the check-box also triggers a noticeable visual manipulation of the regular send button; see Fig. 2.

Sending protected emails requires CherAmi to be able to access the sender's OSN account. Hence, upon installation, CherAmi will ask Alice to log into her OSN account and authorize CherAmi. By design, Alice will be able to send an encrypted email to Bob, only if Bob is among Alice's OSN contacts. OSN relationship models differ from site to site; however, we use the common word *friendship* to refer to an OSN relationship that requires a direct connection between Alice and Bob; Alice must be able to verify Bob's OSN profile and send him a private message through the OSN site. Alice's OSN friends' list can be searched for Bob to check if they are friends, using e.g., ID_B or Bob's real name, and Bob's OSN profile must be verified by Alice for the first protected email.

When Alice indicates that she has finished the message composition (e.g., by hitting the *Send via CherAmi* button), the CherAmi add-on performs the actions as listed in Protocol 1 (under "Sender-side actions").

We do not encrypt the email subject line ($Subject$), similar to PGP. Users may decide on opening an email based on its $Subject$, or later search emails using keywords from subject lines. However, we integrity-protect $Subject$ and other selected header items.

Messages sent by CherAmi are wrapped inside a header (Hdr) and a footer (Ftr), and different parts of OSN messages and email content are separated using markers (Mrk_k and Mrk_m). It makes the receiver (Bob) aware of the use of CherAmi, and also allows the add-on to detect and process protected emails in a straightforward way. One simple yet appropriate choice for such markers is a hash (#) character, which allows easy concatenation of base64-encoded strings.

Decrypting a received email. Bob first receives the encrypted version of the email (i.e., C); see step 3 in Fig. 1. Next, the add-on automatically verifies the email and retrieves the encryption key to decrypt the email content. To avoid privacy breaches in public places due to automatic decryption, a configuration option may be added to the add-on to ask for confirmation from Bob before decrypting the email. For verification, the add-on performs the actions as listed in Protocol 1 (under "Receiver-side actions").

Replying to an email. When replying to an encrypted email from Alice, Bob must encrypt his message. To be on the safe side, we assume that Alice prefers a confidential response to her encrypted email. Hence, the confidential check-box is turned on by default for replies to protected emails. The same steps are then followed as for composing and sending an encrypted email. Alice can choose to encrypt her reply to an unencrypted email.

3.2 Variants

We outline several variants below. We have not implemented these variants in our current prototype.

(a) Integrity checks for unconnected users. If Alice and Bob have no OSN relationships, they can still achieve content integrity protection, but no identity verification or message confidentiality. Alice can publicly post the hash of an email, and Bob (or anyone with access to the email content) can verify the content. Alice should include her OSN ID in the email content, so that Bob can easily locate her profile and check the hash value. Both Alice and Bob must be notified that only content verification is provided. Bob is also asked to manually verify Alice's identity by reviewing her OSN profile.

(b) Sharing keys through other channels. OSNs provide an easy way for sharing per-email keys; however, other channels can be used instead. If a user's contact list with phone numbers is available to a CherAmi add-on (e.g., when used from a smartphone), the keys can be sent via SMSes to the recipient. Note that assumptions regarding two independent channels being operated under different jurisdictions should not be violated. For example, the local mobile carrier must not back up its databases of SMS messages using cloud storage providers that fall under the same jurisdiction as the mail provider. Contact lists from instant messaging applications may also be used for key transport. The CherAmi add-on must be able to automate the key extraction from these secondary channels; i.e., users cannot be expected to manually input key materials.

Protocol 1 CherAmi add-on steps for encrypted emails

Sender-side actions:

1. Generate two symmetric keys K_m and K_h, both of which are specific to the current email.
2. Securely publish $K_m||Mrk_k||K_h$ on the OSN site using Alice's account such that it is instantly accessible *only* to Bob (e.g., as a direct private message in Twitter to his account); see step 1 in Fig. 1. Obtain M_{id} from the OSN site. In the case of multiple recipients, several private messages are sent and M_{id} includes all message identifiers.
3. Compute $C_m = E_{K_m}(M)$.
4. Compute $C_h = MAC_{K_h}(ID_A||Subject||ID_B)$. In the case of multiple recipients, ID_B includes all addresses from `To:` and `CC:` email header fields; `Bcc:` addresses are excluded as they are not available to all recipients.
5. Create $C = Hdr(M_{id})||C_m||Mrk_m||C_h||Ftr$ and replace message body by C.
6. Send the processed email message C with all header parameters to ID_B via the email service provider; see step 2 in Fig. 1. In the case of multiple recipients, C will be sent to every recipient in the list. The user-to-OSN/email channels must be protected (e.g., via SSL), as explained under "Channel and host security" in Section 2.

Receiver-side actions:

1. Retrieve C along with the email header from the email provider; see step 3 in Fig. 1.
2. Tokenize C to obtain $Hdr(M_{id})$, C_m and C_h.
3. Use M_{id} to retrieve the private message sent from Alice via the OSN site, and tokenize the message to obtain K_m and K_h; see step 4 in Fig. 1.
4. Compute $MAC_{K_h}(ID_A||Subject||ID_B)$ and compare it to C_h. A match verifies Alice's identity and the integrity of the header information. A mismatch implies either the identity of Alice cannot be verified, integrity of the header parameters is lost, or both. In this case, warn Bob about the verification failure through the user interface.
5. Decrypt C_m to obtain M, and replace the message body with M. A successful decryption verifies Alice's identity and the integrity of the received email (due to the use of authenticated encryption).

(c) Group emails. A group email address can comprise an unlimited number of email users. Group emails may be supported for integrity and origin authentication, if the email group is also represented as an OSN entity (e.g., Twitter lists). Each group member must be connected to the OSN group whose posts are only made available to its members. Note that confidentiality may be difficult to maintain when there are many users in a group.

4. IMPLEMENTATION

Our implementation is based on a Thunderbird add-on, registered as a Twitter app. The prototype highlights challenges of implementing a rather simple design on top of existing email/OSN services. These challenges also show why real-world implementation is non-trivial, compared to a stand-alone, proof-of-concept implementation. A stand-alone prototype with a specific email client and a custom OSN (e.g., managed by a service run by us) could have reduced our efforts. However, we believe that there is little to no chance of such proof-of-concept implementations being used in practice.

The email client add-on approach allows users the freedom in choosing the email provider, as differences between various email providers are abstracted away behind well-supported and standard IMAP, POP3 and SMTP protocols. Our choice of Twitter as the OSN provider is affected by several factors. First, Twitter is one of the most popular OSN sites today, allowing easier adoption of the solution for many users. Second, Twitter offers a relatively stable and easy to use REST API, which can be easily consumed from an external app, including a Javascript-based Thunderbird extension. Third, Twitter apps are allowed to use the API for sending and receiving private messages among Twitter users through an SSL-protected channel. This is a strict requirement for our proposal, preventing us from leveraging certain popular OSN sites, including Facebook, as the key-transfer channel.[10] Based on these factors, we have chosen Twitter. However, there are certain security considerations and challenges, which should be taken account when such a solution is implemented using Twitter. These specific issues are discussed in Section 4.1.

4.1 Thunderbird Add-on

For encryption support, we use the Stanford Javascript Crypto Library (SJCL [50]). CherAmi requires a cryptographically secure random key for each confidential email. Although low entropy issues in the SJCL random number generator[11] are fixed in its latest version, SJCL heavily relies on mouse movements as the source of entropy. While mouse movements seem like a good source of entropy, the approach has a clear issue: in the case of insufficient mouse movements immediately before the user chooses to send a confidential email, adequate entropy of the generated keys cannot be guaranteed. As a result, we use W3C web crypto APIs (www.w3.org/TR/WebCryptoAPI/), available in Thunderbird version 21 and later to generate random numbers, and use these numbers as entropy sources for the SJCL random key generator. For authenticated encryption, we use AES-256 in the CCM mode [14].

The add-on supports both plaintext and HTML/rich content. An email message encrypted by CherAmi is wrapped by a special header and footer, allowing the add-on on the recipient's side to distinguish it from regular email messages. The header includes sufficient information for the receiver add-on to access the per-email key-phrase sent as a direct private message to the recipient's Twitter account.

[10]As of Apr. 2015, new Facebook apps are forced to use the Graph API version 2. This API does not allow sending private messages (see: https://developers.facebook.com/docs/graph-api/reference/v2.3/message). Currently, Facebook only supports sending messages using explicit message dialogs.

[11]https://github.com/bitwiseshiftleft/sjcl/issues/77

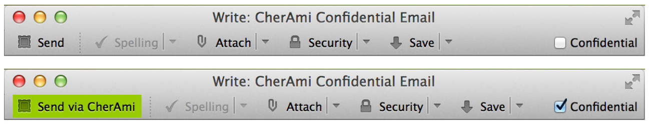
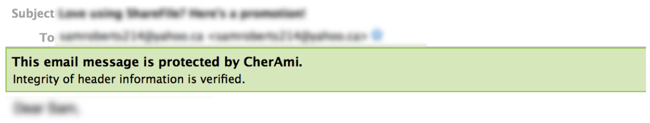


Figure 2: Compose toolbar modifications



This email message is protected by CherAmi.
Integrity of header information is verified.

Figure 3: CherAmi's notification bar

The wrapped part of the email body contains the base64-encoded ciphertext.

User interface changes. We integrate CherAmi's functionality into the existing Thunderbird application while keeping visual modifications to a minimum. CherAmi adds a special *Confidential* check-box in the email compose window (placed on the right side of the toolbar). Once this check-box is turned on, the regular *Send* button will be replaced by a *Send via CherAmi* button; see Fig. 2.

A user may simply open a compose window, write a confidential message and click on the regular send button by mistake; cf. "the barn door property" [61]. To mitigate this concern, we also add a *Secure Write* button to the main Thunderbird toolbar, which sits next to the regular *Write* button; *Secure Write* opens a compose window with the confidentiality check-box turned on by default.

On the receiver's side, if a protected message is detected by the add-on, a notification bar is displayed above the message content area. This notification bar informs the user about the message being protected by CherAmi. This approach is chosen in consistence with the regular Thunderbird interface, which informs the user about spam, remote contents and similar facts through notification bars.

Sending an encrypted email. Once the user clicks on the *Send via CherAmi* button, the following steps are executed by the add-on to send an encrypted email:

1. From the recipients list, the add-on identifies those recipients' email addresses (if any) that are yet to be associated to a Twitter follower.
2. For every new recipient email address, a dialog box with the user's Twitter follower list is shown; the user then selects one of his Twitter followers to be associated with the corresponding recipient email address. We fetch the follower list once, and allow the user to search in the list without making additional calls to the Twitter API. New recipient addresses and the associated Twitter follower names are saved for later use.
3. An array of 32-bit integers obtained from the W3C web crypto API random number generator are used as the source of entropy for SJCL, which is afterwards used to create two random key-phrases. SJCL uses PBKDF2 to derive symmetric encryption keys from these key-phrases. One of these keys will be used for encryption, and the other one will be used for computing the MAC of header parameters.
4. The Twitter direct message API is called to send the generated key-phrases to every Twitter follower with an associated email address in the recipients list. Upon every API call, Twitter returns the ID of the newly created direct message. This ID is extracted from the response and added to the list of message IDs.
5. The content of the email is encrypted using the first generated key, combined with the PBKDF2 process parameters inside a Javascript object, serialized into a JSON string, and then encoded to base64.
6. Step 5 is repeated for each attachment, encrypting and encoding the contents of each attached file. Encrypted attachments are stored in a temporary disk directory, and automatically deleted upon closing Thunderbird.
7. A message authentication code (MAC) of the message subject line along with the list of recipients' email addresses is computed using the second key generated in step 3.
8. A wrapping header is generated by putting the list of message IDs into a fixed template. The base64-encoded ciphertext (step 5) is concatenated with the calculated MAC (step 7) along with a special marker; the result of this operation is wrapped between the header and a fixed footer.
9. The content of the composed email message is replaced by the wrapped string, and the regular send command, which is normally called upon clicking the send button, is issued to send the encrypted message.

Receiving an encrypted email. At the receiver's end, the add-on detects a confidential email by matching the content of the received email against a regular expression representing the wrapper template. After displaying the notification bar, the add-on performs the following steps:

1. The add-on extracts the list of message IDs from the wrapping header. Each message ID will be used to call Twitter's direct message API to retrieve the corresponding key-phrases.
2. The wrapped part of the email content is base64 decoded to reveal PBKDF2 parameters, ciphertext, and the message authentication code generated at the sender's side.
3. The first key-phrase and the PBKDF2 parameters are passed to SJCL to regenerate the encryption key. The key is used to decrypt the ciphertext, and the message content is replaced by the revealed plaintext.
4. If the user selects to open any attached files, CherAmi downloads the encrypted attachment from the mail server, decrypts it using the symmetric encryption key, and stores the decrypted file in a temporary location on disk. The decrypted version is then opened by an appropriate application installed on the user's machine. Decrypted files are automatically deleted when Thunderbird is closed.
5. A MAC of the message subject line along with the list of recipients' email addresses is computed using the second key derived in step 4. If the calculated MAC does not match the received MAC, the user is notified by changing the color and description text of CherAmi's notification bar.

4.2 Twitter Integration

To access Twitter on behalf of a user, CherAmi asks the user to authorize it on Twitter.com; see Fig. 4. Upon clicking a button in the authorization request page, a so called

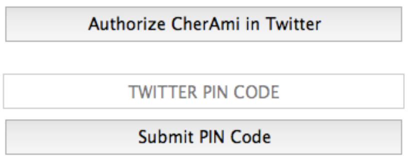
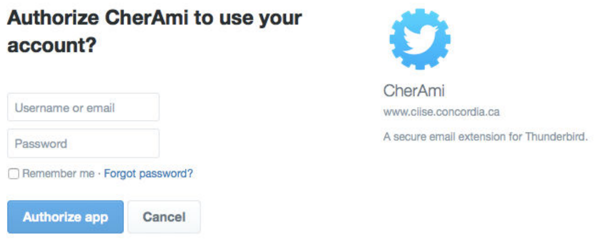


Figure 4: Initiating Twitter authorization


Authorize CherAmi to use your account?

CherAmi
www.ciise.concordia.ca
A secure email extension for Thunderbird.

Figure 5: Twitter authorization for CherAmi

"content tab" in Thunderbird is opened, which allows the user to authorize CherAmi in Twitter; see Fig. 5. The authorization process is a PIN-based OAuth 1.1 flow,[12] in which no HTTP redirects are performed and the user is required to provide the application with a Twitter-issued PIN code. CherAmi uses the PIN code to obtain access tokens, which allow it to access Twitter on behalf of the user. According to Twitter OAuth documentation, access tokens are not expired as of June 2013.[13] Thus, apps like CherAmi are not required to deal with expired tokens or access token refreshment procedures. Twitter supports three different levels of permissions for apps.[14] To send and receive direct messages, CherAmi requires the following permissions: "read, write and direct messages".

5. ATTACKS AND LIMITATIONS

In this section, we discuss several attacks against the CherAmi mechanism; for limitations of the security through hostility paradigm, see Section 2. For threats described in items (b), (c), and (e), an obvious solution is to keep the CherAmi email address separate from the one used in the OSN account.

(a) Information leakage. With CherAmi, the OSN service receives key values for every encrypted email, including identities of the communicating parties. The OSN provider also learns every time an encrypted email is accessed (unless the retrieved key values are stored locally). Although the email content remains protected, the OSN provider now has access to the communication patterns of protected emails between two users. To restrict such leakage, the sender's add-on may post bogus key values; on the receiver side, the add-on may retrieve multiple key values, and then use/ignore these values as appropriate.

(b) Email as account recovery. Usually, OSN providers rely on the user's email account for password recovery. Thus, if a user's email account is compromised, it is potentially trivial to also compromise the OSN account. Therefore, we recommend that CherAmi users be careful with their email accounts (e.g., logging in only from user-owned devices), and use alternative password recovery options (e.g., via SMS). Another obvious solution is to keep the CherAmi email address separate from the one used in the OSN account.

(c) OSN email notifications. OSN providers might send email notifications for events such as receiving a new message, e.g., Twitter notifications for direct messages.[15] Notifications for CherAmi messages must be disabled; otherwise, keys and hashes are sent to the user's email address, allowing the email provider access to confidential email content.

(d) Compromised email accounts. If a sender's email account is compromised, but not the OSN account, several attacks can be considered. An attacker will be unable to decrypt encrypted messages without access to the victim's OSN account; note that, we do not consider brute-forcing a random symmetric AES key. The attacker also cannot send any new protected emails from the compromised account (requires the OSN account access). The attacker can also resend a captured CherAmi-processed email without even compromising the sender's email account. However, any changes to the header fields will be identified by CherAmi at the receiver's side due to failure in MAC verification.

(e) Risks from friend finders. Twitter and other OSN sites commonly ask users to log into their email accounts to send automatic invitations to email contacts. Even if OSN sites claim that they would not access anything beyond email contacts, or would not store the email password, sharing email passwords must be avoided. Otherwise, confidential emails protected by CherAmi may be compromised by OSN sites; users may be warned about this risk by the add-on.

(f) Compromised OSN accounts. An attacker who has compromised the victim's (Alice) OSN account has access to all stored key values. The attacker can also monitor the OSN account for new keys as they are posted. If he has access to encrypted emails, the attacker can now read confidential emails. He can also launch impersonation attacks as follows. The attacker creates an encrypted email message for Bob, impersonating Alice (i.e., `From` $= ID_A$), and publishes the key on Alice's OSN account. Now Bob can decrypt the attacker's message, and assume that it came from Alice. The attacker can also delete all stored keys. This will not affect the decryption and validation of past emails, if local copies of all keys are kept.

(g) No trusted third-parties. We introduce no third parties in our design; users are also not required to trust the CherAmi developers. CherAmi implementation is open-source. We do not run any service for users, or collect *any* information from users. Such a design choice may help user-acceptance, as a user's email and OSN accounts could be extremely privacy-sensitive. For the downside of this choice, see Section 4.2.

(h) Rogue add-ons and email content. Attackers may succeed in getting unauthorized access to users' email account in case of successful phishing attacks. An attacker may

[12]https://dev.twitter.com/docs/auth/pin-based-authorization
[13]https://dev.twitter.com/docs/auth/oauth/faq
[14]https://dev.twitter.com/docs/application-permission-model
[15]https://support.twitter.com/articles/127860-updating-your-email-preferences

mislead a victim into installing a malicious version of the CherAmi add-on, which, for instance, transparently sends copies of plaintext email to the attacker. Like installing any other software, including Mozilla add-ons, users must be careful with software sources they trust (albeit difficult for everyday users). Thunderbird also warns users when they attempt to install add-ons from unknown sources.

6. DISCUSSION ON USABILITY AND DEPLOYMENT

Below we provide an analytical usability and deployment analysis of CherAmi. We have tested the current implementation within our group, mainly to check functionality and UI issues. The add-on worked as expected. However, no formal user studies have been performed yet. Usability issues discussed below could serve as a starting point for future user testing.

Deployment issues with email and OSN providers.

(a) Email providers lose access to confidential messages, and thus, cannot directly benefit from content-aware advertising. However, generic ads (or ads based on the subject line alone) can still be served. As only selected messages will be confidentiality-protected, revenues for ad-supported email services are unlikely to suffer.

(b) OSN providers may observe only minor changes to the number of posted messages, even if CherAmi is adopted widely. Each protected email will result in additional content posted to OSNs; however, the size of each post is relatively small (e.g., tens of bytes). Note that, as of July 2015, Twitter limits the number of direct messages to 1000/day.[16] Although the global email volume per day is large, CherAmi may be used only for a small fraction of selected messages.

(c) OSN providers will also receive more queries due to CherAmi retrieving friends list and posting private messages. Queries are issued when sending and receiving each protected email. Server-side costs for these queries would be non-trivial; the average number of friends is expected to be moderate (e.g., 208 in Twitter as of Oct. 11, 2012[17]); however, the number of posted messages from CherAmi and regular OSN usage would likely grow over time (unless old posts are gradually deleted). Note that modern OSN providers are apparently well-equipped to handle such loads, as evident from their support for thousands of OSN-specific apps that make extensive use of such queries.

Usability issues.

(a) Users are required to understand the security through hostility paradigm to some extent so that they can select appropriate cloud services. For example, users must not select Twitter with Gmail (both under US jurisdiction); Gmail users must select a non-US OSN service (e.g., Russian VK, currently not integrated in CherAmi). Users will be guided through the CherAmi user interface to avoid the selection of certain service combinations (e.g., Gmail with Twitter); however, at the end, users may still need to use their judgement to verify that the selected services are hostile enough for their need (i.e., not to rely fully on the CherAmi UI).

[16]https://support.twitter.com/articles/15364-twitter-limits-api-updates-and-following

[17]http://expandedramblings.com/index.php/march-2013-by-the-numbers-a-few-amazing-twitter-stats/

Users may need to open new accounts in such foreign OSN services, and build their friends list (possibly consisting of selected contacts who want to share confidential messages); this list is expected to be smaller than their regular OSN friend list. Another way for Gmail/Twitter users is to use QQ mail or Mail.ru for all their confidential messages.

We accept that breaking the current user habit of relying on few large service providers, for the sake of privacy gain, may not be so easy, given the wide-spread misconception of "I've got nothing to hide" [47] among users; (but see [10, 12]). On the other hand, many *privacy-agnostic* but *peace-loving* users may embrace CherAmi and related mechanisms that exploit the hostility paradigm; adoption of such mechanisms may force enemy states to cooperate for effective surveillance, and thereby reduce their differences and active hostilities, and indirectly help achieve world peace and stability.

(b) Assume that a sender, Alice, is already logged into her OSN account; she also identified Bob's OSN profile (the receiver) to the CherAmi add-on, when she first sent a protected email to Bob. Subsequently, Alice only needs to turn on the *Confidential* check-box for confidential emails. Bob can open an email as usual; sender authentication, message verification, and decryption are performed automatically. However, to benefit from CherAmi protections, Bob must check UI notification messages as provided by the add-on.

(c) Users must explicitly select confidentiality and integrity protections for their messages. This allows users to control how their messages are protected. However, users may mistakenly send out sensitive information unprotected. This risk is unavoidable as long as we cannot encrypt all emails by default, which may break email communications in many scenarios (e.g., emails sent to OSN-unconnected receivers).

(d) Searching for email content is a useful feature for many users, and could be even more efficient than organizing emails through complex rules (see e.g., [60]). However, keyword searches within server-stored encrypted emails is not supported for now. Searches on locally-stored decrypted emails can be easily supported (e.g., by saving the decrypted versions, or the keys). Note that search in encrypted data is an active research area; notable example solutions to date include: Song et a. [48], EDESE (easily-deployable efficiently-searchable symmetric encryption scheme [29]).

(e) Users must be comfortable with installing a Thunderbird add-on and authorizing a Twitter app as required by our current implementation. Additionally, to prevent certain attacks (e.g., items (b), (c) in Section 5) users must configure Twitter account options (e.g., change the password recovery option from email to SMS). Our solution targets users who already have an email and OSN accounts, and use these on a regular basis. Therefore, we believe that such users already possess the level of technical expertise required by CherAmi. Also, the user is asked to perform a (one-time) manual account mapping between an email address and the corresponding OSN account, similar to the existing practice of sending/accepting an OSN friend request.

7. RELATED WORK AND COMPARISON

We could not locate any past work directly related to the security through hostility paradigm. Except for the hostility requirement, other closely related concepts include: key-splitting [16, 5], secret-sharing [45, 4], and multi-channel secret distribution [37, 8]. In terms of preferentially using/avoiding certain regions and Autonomous Systems when building Tor circuits, the LASTor [2] client is relevant. The use of competing or favorable jurisdictions, often referred as *jurisdictional arbitrage/shopping*, is not new to legal and finance communities (see e.g., [41]). Cyber criminals also reportedly exploit jurisdictional differences or weak laws for launching attacks (see e.g., [34, 27]). To ensure physical safety, traditionally, activists and criminals often take shelter in countries which may not extradite them to accusing countries—e.g., the asylum of Edward Snowden in Russia. In our approach, we leverage hostility between nations to enhance privacy protections for everyday users.

Related to CherAmi, in terms of email confidentiality, numerous proposals exist. Below, we discuss a few representative schemes, but exclude enterprise solutions as they are unsuitable for zero-cost mass deployment. We also do not discuss recent attempts such as Google end-to-end[18] and Keybase,[19] both of which are still in a proposal phase; however, when realized, they may also face similar key management issues as in existing PGP-based proposals.

Waterhouse [28] proposes to use OSN sites such as Facebook to distribute long-term public-keys, by posting them on OSN profiles and leveraging existing OSN connections between users. For intuitive identity verification, Waterhouse suggests displaying a sender's OSN photo with each email (cf. Facemail [30]). As the sender's photo alone cannot guarantee trustworthiness of the OSN profile owner, the use of a web-of-trust model is also suggested; only public keys of senders with at least n friends in common with the receiver are accepted. The burden of managing private keys remains on end users. No implementation of this proposal is available (as far as we are aware of).

Stream [18] is a POP and SMTP proxy that sits between the email client and server, automatically encrypting emails when the receiver's public key is available (opportunistic encryption). If no key pairs are found, public/private keys are generated and stored on the fly; the sender's email address is used as an index to locate the associated key pair in the key database. Stream offers opportunistic key distribution by signing each recipient's public key and adding it to each email's header. At the receiver side, the POP proxy verifies the sender's public key, decrypts and delivers the email. Stream eases the burden of key management but requires users to trust its proxy servers with private keys; such a trust model is particularly unsuitable for webmail providers.

To reduce user involvement, *STEED* [26] proposes significant changes to email providers and Mail User Agents (MUAs). MUAs would automatically generate (public, private) key pairs, or self-signed certificates, each time a new email account is created, and perform opportunistic encryption. Key distribution is done through the DNS server of an email provider, and a trust-upon-first-contact model is proposed, similar to SSH trust-upon-first-use. Users are still responsible for managing their private keys.

Aquinas [8] employs symmetric encryption with per-email keys, and thus avoids several key management issues. An implementation of Aquinas as an open-source Java applet enables confidentiality through AES encryption, deniability of exchanged secret messages through steganography, and message integrity using MAC. Keys and encrypted messages are split, and transmitted separately through competitor email providers. A malicious ISP may collect all key/message shares and retrieve the message, when user to email channels are not SSL-protected. To restrict this attack, bogus message shares are also communicated; e.g., 20 out of 40 shares may be used for the actual message. The ISP now sees all 40 shares, but does not know which ones would construct the real message. The sender and receiver must communicate an initial secret that will be used to determine the shares for the real message; this secret should be established through an out-of-band channel (e.g., phone).

TrustSplit [17] proposes the confidentiality as a service (CaaS) paradigm, and splits the trust between a traditional cloud provider (e.g., Gmail, Dropbox), and the newly introduced CaaS provider(s). To protect user data, multiple layers of commutative encryption are used, which can be added/removed from user data in an arbitrary order. TrustSplit requires third parties to run CaaS servers; users must also register with these services.

SPEmail [37] uses secret sharing and linguistic steganography to provide confidentiality for webmails. Each message is divided into two shares. After encoding them through a form of text steganography, secret shares are delivered via two different webmail providers. No sender authentication or message integrity can be provided.

Brief comparison. We do not require any new user-level secrets; our key management is completely transparent to users. We use the familiar OSN trust relationships, and avoid (largely-failed) past trust models, e.g., certificate/web-of-trust (as used in e.g., S/MIME and PGP, respectively). Compared to secret-sharing proposals that use multiple channels, we do not require users to create and distribute multiple email accounts to transfer shares. Also, the use of per-email encryption keys in CherAmi enables forward secrecy to some extent; e.g., there is no use of long-term encryption keys and the disclosure of an encryption key compromises only the email protected by that particular key.

8. CONCLUDING REMARKS

Internet has enabled people from hostile territories to access cloud-based services from any jurisdiction. To address long-lasting key management issues in cryptography, we introduce the *security through hostility* paradigm, by storing keys and encrypted content to services hosted from mutually non-cooperating jurisdictions. We argue that this paradigm can enable end-to-end encryption between users, without any user-managed keys. Obvious usability concerns include: users must understand how to select services (i.e., jurisdictional knowledge is required), and they may need to open new accounts and convince their (selected) contacts to use additional services. For example, US Twitter users must use an email provider hosted from non-cooperating countries (e.g., QQ.com or Mail.ru); to use Gmail, US users must use an OSN from non-cooperating countries (e.g., VK.com). In a post-Snowden world, convincing privacy-concerned users to take this extra step of opening new accounts may not be too difficult; users are already reportedly showing more interest in privacy tools such as Tor and PGP (see e.g., [10, 12]).

Obviously, hidden collaboration between seemingly hostile nations will fail any privacy-gain via the hostility paradigm.

[18]A proposed Chrome extension based on OpenPGP: https://github.com/google/end-to-end.

[19]A proposed public directory for auditable public keys: https://keybase.io.

In such cases, and for military and enterprise-grade security, traditional end-to-end security solutions must be used (S/MIME, PGP, high-entropy shared secret via personal meeting). Wide-scale adoption of tools leveraging the hostility paradigm can bring hostile nations closer and thereby, promote peace and stability world-wide. Thus, the paradigm can potentially bear positive results in both cases: hostility leads to privacy gain, and cooperation enables peace (a rare *win-win* situation for a security paradigm).

As a concrete realization of the security through hostility paradigm, we propose and implement CherAmi, to enable end-to-end email encryption. Beyond email, our key transport mechanism may enable privacy protection for additional user-to-user data communication services (e.g., encrypted IMs, file sharing via public cloud). CherAmi does not require any server-side modifications, and thus can be immediately deployed. Our current implementation supports encrypted email between users who are also connected via Twitter (as followers); any email provider can be used, as long as the email provider is hosted from a jurisdiction unfriendly towards the U.S. administration. We highly encourage readers to try out our add-on at: https://madiba.encs.concordia.ca/software.html.

9. ACKNOWLEDGEMENTS

Comments and suggestions from the following helped formulate and refine ideas presented in this paper: our shepherd Matt Bishop, anonymous NSPW2015 and ACSAC2014 reviewers, attendees of NSPW 2015, and members of Concordia's Madiba Security Research Group; we thank them all. We also acknowledge Anil Somayaji for his help with the paper's title. The first author is supported in part by an NSERC Discovery Grant.

References

[1] H. Abelson, R. Anderson, S. M. Bellovin, J. Benaloh, M. Blaze, W. Diffie, J. Gilmore, M. Green, S. Landau, P. G. Neumann, R. L. Rivest, J. I. Schiller, B. Schneier, M. Specter, and D. J. Weitzner. Keys under doormats: Mandating insecurity by requiring government access to all data and communications. Technical Report MIT-CSAIL-TR-2015-026, Computer Science and Artificial Intelligence Laboratory, MIT, July 2015. http://hdl.handle.net/1721.1/97690.

[2] M. Akhoondi, C. Yu, and H. V. Madhyastha. LASTor: A low-latency AS-aware Tor client. In *IEEE Symposium on Security and Privacy*, San Francisco, CA, USA, May 2012.

[3] R. J. Aldrich. Dangerous liaisons: post-September 11 intelligence alliances. *Harvard International Review*, 24(3):49–54, Sept. 2002.

[4] G. R. Blakley. Safeguarding cryptographic keys. In *AFIPS National Computer Conference (NCC'79)*, New York City, NY, USA, June 1979.

[5] M. Blaze. Key escrow from a safe distance: Looking back at the Clipper Chip. In *Annual Computer Security Applications Conference (ACSAC'11)*, Orlando, FL, USA, Dec. 2011. Keynote paper.

[6] Bloomberg.com. U.S. charges five Chinese military officers with spying. News article (May 19, 2014). http://www.bloomberg.com/news/2014-05-19/u-s-said-to-charge-chinese-military-officers-with-online-spying.html.

[7] Y. Boshmaf, I. Muslukhov, K. Beznosov, and M. Ripeanu. The socialbot network: when bots socialize for fame and money. In *ACSAC'11*, Orlando, FL, USA, Dec. 2011.

[8] K. Butler, P. Traynor, W. Enck, J. Plasterr, and P. McDaniel. Privacy preserving web-based email. In *ICISS'06*, Kolkata, India, Dec. 2006.

[9] Q. Cao, M. Sirivianos, X. Yang, and T. Pregueiro. Aiding the detection of fake accounts in large scale social online services. In *USENIX NSDI'12*, San Jose, CA, USA, 2012.

[10] ComputerWorld.com. Users flock to anonymizing services after NSA snooping reports. News article (Oct. 10, 2013). http://www.computerworld.com/s/article/9243141/Users_flock_to_anonymizing_services_after_NSA_snooping_reports.

[11] Cryptome.org. Mining PGP key servers. Mailing list discussion (July 21, 2013). http://cryptome.org/2013/07/mining-pgp-keyservers.htm.

[12] Dailydot.com. After Snowden leaks, daily adoption rate of PGP encryption triples. News article (July 31, 2013). http://www.dailydot.com/news/pgp-encryption-snowden-prism-nsa/.

[13] D. Dolev and A. C. Yao. On the security of public key protocols. *IEEE Transactions on Information Theory*, 29(2):198–208, Mar. 1983.

[14] M. Dworkin. Recommendation for block cipher modes of operation: The CCM mode for authentication and confidentiality (NIST SP 800-38C), May 2004.

[15] M. Egele, G. Stringhini, C. Kruegel, and G. Vigna. COMPA: Detecting compromised accounts on social networks. In *NDSS'13*, San Diego, CA, USA, 2013.

[16] EPIC.org. Clipper encryption – AT&T telephone security device model 3600. Secret briefing document obtained by EPIC by using the US Freedom of Information Act (Feb. 9, 1993), https://epic.org/crypto/clipper/foia/att3600_2_9_93.html.

[17] S. Fahl, M. Harbach, T. Muders, and M. Smith. TrustSplit: Usable confidentiality for social network messaging. In *ACM Hypertext'12*, Milwaukee, WI, USA, 2012.

[18] S. L. Garfinkel. Enabling email confidentiality through the use of opportunistic encryption. In *Conference on Digital Government Research*, Boston, USA, 2003.

[19] S. L. Garfinkel and R. C. Miller. Johnny 2: A user test of key continuity management with S/MIME and Outlook Express. In *SOUPS'05*, Pittsburgh, PA, USA, July 2005.

[20] R. Geambasu, T. Kohno, A. Levy, and H. M. Levy. Vanish: Increasing data privacy with self-destructing data. In *USENIX Security Symposium*, Montreal, Canada, Aug. 2009.

[21] M. Green. The many flaws of Dual_EC_DRBG. Blog article (Sept. 18, 2013). http://blog.cryptographyengineering.com/2013/09/the-many-flaws-of-dualecdrbg.html.

[22] E. Grosse. Security warnings for suspected state-sponsored attacks. Google online security blog (June 5, 2012). http://googleonlinesecurity.blogspot.ca/2012/06/security-warnings-for-suspected-state.html.

[23] J. V. Hoboken, A. Arnbak, and N. V. Eijk. Cloud computing in higher education and research institutions

and the USA Patriot Act, Nov. 2012. http://ssrn.com/abstract=2181534.

[24] ITWorld.com. Facebook's 'man in the middle' attack on our data. News article (Feb. 5, 2012). http://www.itworld.com/it-managementstrategy/247344/facebooks-man-middle-attack-our-data.

[25] L. H. Keeley. *War Before Civilization: The Myth of the Peaceful Savage*. Oxford University Press, 1996.

[26] W. Koch and M. Brinkmann. STEED – usable end-to-end encryption, Oct. 2011. http://g10code.com/steed.html.

[27] N. Kshetri. Pattern of global cyber war and crime: A conceptual framework. *Journal of International Management*, 11(4):541–562, Dec. 2005.

[28] A. P. Lambert, S. M. Bezek, and K. G. Karahalios. Waterhouse: enabling secure e-mail with social networking. In *CHI'09*, Boston, MA, USA, Apr. 2009.

[29] B. Lau, P. H. Chung, C. Song, Y. Jang, W. Lee, and A. Boldyreva. Mimesis aegis: A mimicry privacy shield - a system's approach to data privacy on public cloud. In *USENIX Security Symposium*, San Diego, CA, USA, Aug. 2014.

[30] E. Lieberman and R. C. Miller. Facemail: showing faces of recipients to prevent misdirected email. In *SOUPS'07*, Pittsburgh, PA, USA, July 2007.

[31] W. Lou. Spread: Enhancing data confidentiality in mobile ad hoc networks. In *IEEE INFOCOM*, 2004.

[32] J. J. Mark. War–ancient history encyclopedia. Online article (Sept. 2, 2009). http://www.ancient.eu.com/war/.

[33] M. Ménard. Hidden cooperation: How nuclear antagonists collaborated on counter-narcotics efforts in Iran from 2007 to 2011. Master's thesis, Université Laval, Québec, Canada, 2014.

[34] G. Mezzour, L. R. Carley, and K. M. Carley. Global mapping of cyber attacks. Technical Report CMU-ISR-14-111, School of Computer Science, Carnegie Mellon University, 2014.

[35] R. M. Needham and M. D. Schroeder. Using encryption for authentication in large networks of computers. *Communications of the ACM*, 21(12):993–999, 1978.

[36] J. A. Obar and A. Clement. Internet surveillance and boomerang routing : A call for Canadian network sovereignty. In *Technology and Emerging Media Track (TEM'13)*, Victoria, BC, Canada, June 2013.

[37] Y. Oren and A. Wool. Perfect privacy for webmail with secret sharing. Technical report, Feb. 2009. http://www.eng.tau.ac.il/~yash/OrenWool-SPEmail.pdf.

[38] P. Pennock. Trusting PGP. *USENIX ;login:*, 38(6):33–35, Dec. 2013.

[39] R. Person. Crouching tiger, hidden jargon: The Sino-Russian strategic partnership. *Stanford Journal of International Relations*, 3(1), 2001.

[40] A. S. Pirouz, V. Rabotka, and M. Mannan. Friendly-Mail: Confidential and verified emails among friends. Technical Report 978331, Spectrum Research Repository, Concordia University, Mar. 2014. http://spectrum.library.concordia.ca/978331/.

[41] G. Poitras. Arbitrage: Historical perspectives. In *Encyclopedia of Quantitative Finance*. John Wiley & Sons, May 2010.

[42] M. D. Ryan. Enhanced certificate transparency and end-to-end encrypted mail. In *NDSS'14*, San Diego, CA, USA, Feb. 2014.

[43] B. Schneier. Attacking Tor: how the NSA targets users' online anonymity. News article (Oct. 4, 2013). http://www.theguardian.com/world/2013/oct/04/tor-attacks-nsa-users-online-anonymity.

[44] B. Schneier. *Applied Cryptography*. John Wiley & Sons, 2 edition, 1996.

[45] A. Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, Nov. 1979.

[46] C. Soghoian and S. Stamm. Certified lies: Detecting and defeating government interception attacks against SSL (short paper). In *Financial Cryptography and Data Security (FC'11)*, St. Lucia, 2011.

[47] D. J. Solove. 'I've got nothing to hide' and other misunderstandings of privacy. *San Diego Law Review*, 44:745–772, July 2007.

[48] D. Song, D. Wagner, and A. Perrig. Practical techniques for searches on encrypted data. In *IEEE Symposium on Security and Privacy*, May 2000.

[49] Spiegel.de. Targeting Huawei: NSA spied on Chinese government and networking firm. News article (Mar. 22, 2014). http://www.spiegel.de/international/world/nsa-spied-on-chinese-government-and-networking-firm-huawei-a-960199.html.

[50] E. Stark, M. Hamburg, and D. Boneh. Symmetric cryptography in javascript. In *ACSAC'09*, Honolulu, HI, USA, 2009. http://crypto.stanford.edu/sjcl/.

[51] TechCrunch.com. Russia moves to ban online services that don't store personal data in Russia. News article (July 2, 2014), http://techcrunch.com/2014/07/02/russia-moves-to-ban-online-services-that-dont-store-personal-data-in-russia/.

[52] TheGuardian.com. The NSA files: How the story unfolded. Guardian news archive (from June 5, 2013). http://www.theguardian.com/world/the-nsa-files.

[53] TheGuardian.com. NSA Prism program slides. Leaked Prism slides (Nov. 1, 2013). http://www.theguardian.com/world/interactive/2013/nov/01/prism-slides-nsa-document.

[54] TheGuardian.com. Revealed: how US and UK spy agencies defeat internet privacy and security. News article (Sept. 6, 2013). http://www.theguardian.com/world/2013/sep/05/nsa-gchq-encryption-codes-security/print.

[55] TheGuardian.com. Secrets, lies and Snowden's email: why I was forced to shut down Lavabit. News article (May 20, 2014). http://www.theguardian.com/commentisfree/2014/may/20/why-did-lavabit-shut-down-snowden-email.

[56] M. Tracy, W. Jansen, K. Scarfone, and J. Butterfield. Guidelines on electronic mail security (NIST SP 800-45 Version 2), Feb. 2007. http://csrc.nist.gov/publications/nistpubs/800-45-version2/SP800-45v2.pdf.

[57] C. Velasco. Cybercrime jurisdiction: past, present and future. *ERA Forum*, pages 1–17, June 2015.

[58] L. von Ahn, M. Blum, N. Hopper, and J. Langford. CAPTCHA: Using hard AI problems for security. In *Eurocrypt*, Warsaw, Poland, May 2003.

[59] WashingtonPost.com. NSA-proof encryption exists. why doesn't anyone use it? Blog article (June 14, 2013).

[60] S. Whittaker, T. Matthews, J. Cerruti, H. Badenes, and J. Tang. Am I wasting my time organizing email? a study of email refinding. In *CHI'11*, Vancouver,

Canada, May 2011.

[61] A. Whitten and J. D. Tygar. Why Johnny can't encrypt: A usability evaluation of PGP 5.0. In *USENIX Security Symposium*, Washington, DC, USA, Aug. 1999.

[62] S. Wolchok, O. S. Hofmann, N. Heninge, E. W. Felten, J. A. Halderman, C. J. Rossbach, B. Waters, and E. Witchel. Defeating Vanish with low-cost sybil attacks against large DHTs. In *NDSS'10*, San Diego, CA, USA, 2010.

[63] F. L. Wong and F. Stajano. Multichannel security protocols. *IEEE Pervasive Computing*, 6(4):31–39, Oct. 2007.

[64] ZDNet.com. Former NSA executive: Snowden leaks caused 'significant disservice' to the Internet. News article (Apr. 24, 2014).