Glypher – A Discourse on User-frendly Email Encryption

Robert Gorrie
McMaster University

I. Introduction

Back in 1999, Alma Whitten & J. D. Tygar of Carnegie Mellon University published an article titled *Why Johnny Can't Encrypt*. The primary concern of the study was the ease with which users could encrypt and decrypt personal emails using the PGP standard. The authors of the paper provided a specific definition for 'usability of security'. This was in lieu of their claim that interface design for security and cryptography software faces significantly different challenges than general-purpose software. Two followup studies (*Why Johnny Still Can't Encrypt* and *Why Johnny Still, Still Can't Encrypt*) have henceforth been released, reporting little to no imporovement in the usability of PGP tools.[1],[2],[3]

Throughout the course of this brief report, we will explore why these studies may have yielded unfavourable results. We will emphasize the importance of information hiding and domain constriction in consumer-grade security software while looking to several examples. Most notably, we will introduce Glypher, a lightweight unix program built to interface with GPG and SMTP, and designed to apply the security design principles that we will shortly explore.

II. A Review of Security Software Usability

All Three of the aforementioned papers reported the results of typical usability studies for different PGP clients. The clients in question were PGP 5.0, PGP 9, and Mailvelope. Each research group measured the time taken to complete different cryptographic procedures. In some cases, test subjects were entirely unable to complete the task. As a result, each group concluded that their respective test clients were not usable for the 'average' computer user.

In the field of human computer interactions and interfaces, there are several golden standards for usability that the industry swears by. Many of these principles were popularized by Don Norman. The design principles outlined by Norman are that of consistency, affordance, feedback, visibility, mapping, and constraints. These golden standards, however, were challenged by Whitten and Tygar in the context of security software. The original *Why Johnny Can't Encrypt* makes the claim that effective security interfaces demand a usability standard different from that of typical consumer software.[4],[1]

A. Whitten & Tygar's Usability Definition

Whitten and Tygar list their priorities for the usability of security software as the following, claiming that any given security software is 'usable' if the users:

1. are reliably made aware of the security tasks that they need to perform;
2. are able to figure out how to successfully perform those tasks;
3. don't make dangerous errors; and
4. are sufficiently comfortable with the interface to continue using it.

Their paper then proceeds to list five properties that they claim *must* be explicitly considered in the production of security software interfaces. These properties are the following:

The Unmotivated User Property

This is the idea that security is generally placed on the back-burner for most users. The typical user does not hold confidentiality or integrity as a central concern when conducting themselves on a computer or over the Internet. This principle dictates that if an interface for security is too difficult, a user may avoid using it altogether. It is worth noting that since the release of this paper (1999) and the conception of a data driven economy, this property has likely changed significantly.

The Abstraction Property

This is the concept that cryptographic primitives and procedures will likely be 'alien and unintuitive' to the average user. This should be factored into interface design by means of abstracting away certain mechanisms, or applying information hiding.

The Lack of Feedback Property

The idea that providing appropriate feedback for security procedures is a difficult task. Adequate consideration of this should result in the property of a system doing what the user 'really wants', in order to bypass potentially innaccurate feedback.

The Barn Door Property

Once a piece of information has been left unprotected, there is no certainty as to whether an attack has gained access to it. In lieu of this, the barn door property is the degree to which a program makes sure the user understands the risk of making mistakes.

The Weakest Link Property

This property emphasizes the importance of every step in a given security procedure. Because a system is only as secure as its weakest link, a user should be guided to care for every step of the procedure with equal concern.

B. The Importance of Information Hiding and Domain Constriction

The previously mentioned studies concerned themselves with what ease a typical user performs different cryptographic operations using Mailvelope, PGP 9, and PGP 5. These operations include, but aren't limited to; encrypting an email, signing an email, creating a key pair, and verifying a digital signature.

This is where the techniques of the previously mentioned studies are ill conceived. It is unreasonable to trust that someone who is unaware of even the most basic cryptographic concepts should be able or be motivated to perform these operations. It is true that through the application of classical user interface design principles we can increase the ease with which certain procedures can be perfomed. However, it is what we *cannot* do through application of these principles wherein the problem lies. We cannot incentivize a given security practice, we cannot imbue knowledge or understanding of the underlying mathematics, and we cannot easily demonstrate the role of a given cryptographic primitive with respect to the protocol in which it resides.

Domain constriction is a rather simple concept, and along with information hiding, provides a case-specific remedy to the previously mentioned difficulties of designing security interfaces. It involves narrowing the functionality of a program in order to narrow the possible outcomes. So now, what might this mean in the security context? This might mean limiting the array of key generation algorithms, it might mean limiting the array of encryption and decryption algorithms, and it might mean limiting the array of signing algorithms. This places low level, technical deciscions in the hands of the software engineers, as opposed to the end users.

In the users mind, he/she wants to accomplish one (seemingly) simple task (secure, confidential email communication, in our example). This is where the concept of domain constriction is crucial. Security specialists decide which protocols and procedures are most optimal for a given security context and provide an interface on a context-to-context basis. In this scenario, security programs are ideally specialized for a particular use – the end user needs only to pick an interface that matches their specific needs.

III. A Brief Overview of Glypher

Our aim with Glypher is to highlight the importance of information hiding and domain constriction in security software. Contrast this, perhaps, with the provision of a general purpose interface for security and cryptography operations, the likes of the aforementioned PGP 9 and PGP 5. Glypher outlines a potential design for an incredibly simplistic and domain specialized security interface. The interface in question is designed for the context of securely sending and receiving confidential emails. The program (though currently under construction,) interfaces with GPG (the GNU Privacy Guard, an open source implementation of the PGP standard) to send encrypted emails.

A. The Usability Design Strategy of Glypher

Of the previously discussed security interface properties, the ones immediately central to the design of Glypher are the abstraction property and the lack of feedback property. Glypher aims to abstract all procedures of encryption, signing, decryption, and signature verification away from the user so that establishing secure communication is as easy as sending a regular email.
In breif, Glypher has two main functionalities. First, it interfaces with GPG to perform cryptographic operations on files. The user types messages into the interface (along with a specified recipient,) and the program sends these files to be encrypted or signed by PGP. Secondly, Glypher packages these files and sends them in an email to the recipient specified.

B. Contemporary Alternatives

There are several readily available products that offer *nearly* the same solution that Glypher aims for. The most notable of these alternatives would be Protonmail. Protonmail is widely promoted and celebrated, though not open source. Tutanota, while less popular, is an open source solution to the same problem. Tutanota offers 1 GB of storage with the ability for upgrade for a premium. Countermail is another service with a non-premium possibility, and provides in-depth security options through OpenPGP or alternatives.

IV. Future Work

It should be made clear that the current implementation of Glypher is not a consumer-grade product. There are several potential security flaws that need to be addressed before Glypher can be safely deployed and reliably used.

There are several functionalities we intend on adding to Glypher before it is ready for deployment. The first is a Key Creation and Key Management system. Currently, the system requires that users add a file for each of their contacts to the ”conts” folder. These files (suffix .key) should contain their contacts' public keys, and be prepended with their email addresses. The program parses the file, sending the desired message to the email address found after encrypting it using the key in the .key file. A way to make this more user-friendly might be to include a file browser, where the user can locate his friends downloaded Glyphs (email with public key) and the system will automatically add them to the correct location.
Signature signing and verifying is another core feature that is over the horizon. The method will be almost identical to how decryption and encryption are performed, and so adding the functionality shouldn't be much of a stress.
The final core functionality yet to be implemented is a direct interfacing with the SMTP protocol. Currently the system uses a very crude (and hardly portable) method of sending emails by means of making system calls. Direct SMTP interfacing will be required to implement real time reception and decryption of messages within the Glypher interface. Moreover,

with direct SMTP action the possibility of adding encrypted attachments could also be realized.

The potential security issues previously mentioned are mostly comprised of potential side channel attacks and end-system security flaws (if someone gains access to system, the confidentiality of your messages is compromised). Measures will be taken to counteract these potential faults. Once solutions to these potential issues (as well as the to-do features mentioned above) are implemented, Glypher will be ready for userbase deployment.

V. Concluding Remarks

We believe that the problem of achieving usable encryption and signature software resides an a failure to narrow the target demographic and establish a linear and predictable program workflow.

References

1) Alma Whitten, J. D. Tygar, *Why Johnny Can't Encrypt*
2) Steve Sheng, Levi Broderick, Colleen Alison Koranda, Jeremy J. Hyland, *Why Johnny Still Can't Encrypt: Evaluating the Usability of Email Encryption Software*
3) Scott Ruoti, Jeff Anderson, Daniel Zappala, Kent Seamons, *Why Johnny Still, Still Can't Encrypt: Evaluating the Usability of a Modern PGP*
4) J. Preece, Y. Rogers, H. Sharp, *Interaction Design: Beyond Human-Computer Interaction*
5) Simson L. Garfinkel, Robert C. Miller, *Johnny 2: A User Test of Key Continuity Management with S/MIME and Outlook Express*
6) Ayo Isaiah, *5 of the Best Secure Email Services for Better Privacy* – https://www.maketecheasier.com/secure-email-services/