CSCI 343 - Final Project

Canyon: A Message-Body Encryption Gadget for Gmail

Brand, Ilona, Nora Hayes, Alex Katz, James Quintana and Tyler Robertson

Spring 2014

CSCI 343 - Final Project

Canyon: A Message-Body Encryption Gadget for Gmail

Abstract

This paper presents Canyon, a new encryption option for IMAP mail messages sent using the Gmail web-browser-enabled mail service by Google. We will begin with a survey of existing encryption options, and reveal the necessity of a new option. We then will provide more detail into the functionality of Canyon, our implemented solution, and its use. We will conclude with a summary of the strengths that distinguish Canyon from the previous options, and the areas we have considered for further development.

Background

The typical email user may not be conscious of the fact that their method of communication is mostly insecure. This sort of user may unknowingly release private information, which is possible to be intercepted by a third party. If the user is aware that email is not entirely secure, they may be dissuaded to use this medium to communicate privately.

The issue of encrypting this widely public method of communication has not been overlooked, as systems using PGP and S/MIME (Secure/Multipurpose Internet Mail Extensions) have existed for almost twenty years now (Adida et al.). The problems with these systems, however, involve poor UIs, where only thirty percent of untrained users can successfully send and decrypt encrypted messages (Sheng et al.)

Fortunately today email encryption systems do exist that are simple enough to use. We examined Chrome extensions Secure Mail for Gmail (by Streak) and SecureGmail (http://www.streak.com/securegmail). These open source applications are embedded into Gmail within the Chrome browser and encrypt messages. We found that the UIs for these applications were easily navigable and usable for untrained users.

We liked how SecureGmail works, but wanted something with multiple browser support. Google Sidebar Gadgets seem to be the solution to this. Unlike extensions, sidebar gadgets display content on the side of the gmail display with any browser. Since we believe people have the right to privacy, this seems to be the best opening for a new PGP implementation.

Canyon

Canyon is our solution to these exigencies. It offers a secure and easy way to communicate encrypted text emails to other users of Canyon. The underlying encryption scheme is inspired by Phil Zimmermann's PGP, although it does not conform to the OpenPGP standard (RFC 4880). There are two basic components to encryption in Canyon: a public-key (asymmetric) encryption method, and a private-key (symmetric) encryption method. The software uses a combination of user-input and saved state to encrypt and decrypt messages that are sent between users.

All encryption methods are implemented in JavaScript. We chose to use RSA for our asymmetric method. RSA is a strong choice because it does not require transmission across previously-existing secure channels to transmit secret keys. However, it is slow to encrypt very large messages; as such, we chose to use 3DES to encrypt the user's message with a randomly generated fixed-size symmetric key, then encrypt that key using the RSA (à la PGP). We chose 3DES due to its strength and speed over large messages. We found and employed JS-Encrypt — Travis Tidwell's wrapper for Tom Wu's JavaScript implementation of RSA (Tidwell). Canyon uses this library for all RSA-specific functions, including public/private key-pair generation and encrypting the 3DES random key. We sourced our 3DES implementation of another library, CryptoJS, which is an open-source collection of several encryption algorithms including 3DES.

The functionality of Canyon is user-dependent, because it saves state unique to each user that is only accessible to that user. Canyon's saved state (for a given user) consists of two pieces of information: a public key and a private key, which together form an RSA key pair. In the future, we hope to expand this functionality to store a user's list of contacts and public keys of those contacts locally as well. These keys are components of the asymmetric encryption method mentioned above; the symmetric system (along with its internal padding scheme) is completely abstracted away and does not make use of Canyon's saved state.

Encrypting a transmission using Canyon requires only that the sender has an accurate RSA public key accessible to them for the intended recipient. An input text message is provided by the sender. First, Canyon generates a random key for 3DES. The provided message text is padded to a length multiple of 64 bits; this padded message is encrypted with 3DES using the randomly generated key. The random key is then encrypted using RSA and the recipient's public key. The encrypted key and encrypted message are output as a securely transmissible message.

Decrypting a received message depends only that the user's private-key and the public key used by the sender comprise a single generated pair (i.e., the recipient did not generate a new key pair but fail to distribute the public component). The received data includes an encrypted 3DES key, which is decrypted using RSA and the user's unique private key. This extracted key is then used with 3DES to decrypt the main message body, which is itself then un-padded using an inverse of the padding scheme mentioned above; this converts the padded 64n-bit (for some integer n) message into the original message that the sender wished to communicate.

Using Canyon

Canyon is a Google sidebar gadget: a platform-independent, portable, embedded application that users can interact with when logged into the Gmail web browser front-end.

Canyon's interface and interactive elements are created from HTML, CSS, and JavaScript. It is best thought of as a webpage that is embedded into Gmail. It is divided into three sections. The first is for encryption, and the second for decryption; the third (optimally used only once) is for generating a unique public/private key pair for your own use.

The first section is for encryption. To access this section, a user clicks the "E" button at the top of the UI. On this page, the user is presented with two text fields. In the first, the user pastes their plaintext message; in the second, the user pastes the public key of the desired recipient. When the user presses “Encrypt”, the full encryption scheme (including 3DES and RSA) is executed, placing cipher text output into the message box at the top of the page, ready-to-email. Note that this output includes headers to demarcate the encrypted message, and the encrypted key, so that the recipient can easily distinguish them.

The second section is for decryption. To access this section, a user clicks the “D” button. The user has received easily separable an encrypted key and a cipher text message. As before, the user is presented with two text fields. In one, the user pastes the ciphertext; in the other, the encrypted 3DES key. When they press “Decrypt”, their stored private key is used to retrieve and display the original message.

The last section, for generating key pairs, is accessible through the “R” button at the top of the gadget. On this page, you can view your RSA key pair. The only interaction that can take place on this page is to generate new keys; however, once a user does this, no messages encrypted for the previous key pair will be readable by that user.

Advantages

Canyon’s most important feature is its cross-platform, cross-browser support. It interacts directly with the Gmail servers. Although its source is hosted on our server, it is generalized in XML and sent by Google to the client's browser. This has the result that it can be run on any browser and machine that support Gmail's front-end interface.

Canyon also has the advantages of being freely available and user-friendly. The setup requires minimal effort, and use case-to-case requires only commonly-known activities such as copy and paste. And while deeper levels of integration than we achieved has its strengths, there is also value in the total transparency that Canyon renders into the steps of its process — very little of the high-level encryption/decryption process is hidden from the user, so while the users aren't burdened with the technical details of the software, they can determine the trustworthiness of the system from an at least slightly informed position.

Disadvantages

The starkest disadvantage to be found in Canyon is our system of trust. By using large third-party libraries such as Crypto-JS and js-encrypt, without thoroughly scanning their contents for problems. We place trust in the developers and the open-source community for this review. Therefore there is a security weakness which would enable any backdoors in these systems to be present in ours. However, we have countered this in one respect by including the current version of these libraries' source in our source, rather than a dynamic link. This means that if a new vulnerability is put into them, we will be protected; but if one is patched, we will have to update as well.

There is also some disadvantage to our storage of the public keys of contacts. We found that while data persistence is possible with sidebar apps, the only native way to achieve it is limited to a total of 2KB of data — which limits us to seven public keys. There is another external API which permits arbitrarily large storage, but the library is meant for social networking and comes with a substantial amount of extra functionality — and we didn't want to include that amount of unused code and potential security flaws in Canyon. Thus we have left it to our users to store the keys in any location they see fit. As such, in order to maintain cross-platformability, we recommend that users store them in a spreadsheet on some cloud-accessible location such as Google Drive.

Further Work

There are some improvements that we would like to implement in the future. We have two directions in which we want to move. First, we hope to integrate more fully into Gmail. In particular, we would like to add functionality for Canyon to compose and send the emails using the encrypted text. Similarly, we'd like to directly scan for emails encrypted using Canyon (possibly using subject lines as identifiers) and decrypt automatically. However, this requires use of Google Apps

Scripts. Apps Scripts are a library that allow Chrome Apps to access Google services directly — such as Gmail and Drive. Though we have evidence to believe that sidebar gadgets are able to use Apps Scripts for similar functionality, we have been unable to find documentation on how to implement it. Additionally, if we can integrate Apps Scripts, we should be able to use them for indefinite data storage in Google Drive. With this, we will be able to abstract away all key-management; the public key-list will be stored there, and the user will never have to directly handle them. Together, we hope that these improvements will solidify Canyon's interface into a unitary, cohesive, and much less cluttered user experience.