SYSADMIN

Trusting PGP

PHIL PENNOCK

Phil is a software engineer at Apcera, provider of the modern enterprise IT platform.
phil.pennock@spodhuis.org

PGP is a great tool, but if you're coming to it now, after this year's NSA revelations, then it's probably not the service you want. In fact, I'll go further: if PGP is being peddled to you as the panacea to the NSA issues, the peddler probably doesn't understand what they're talking about.

In all security decisions, you should decide what you're trying to protect and from whom. Additionally, you should decide how much the protection is worth to you. Only once you've done this, can you decide which attributes (confidentiality, authenticity, etc.) you need and what tradeoffs are worth it.

For various good reasons, I run my own mail service that serves only two people; for various other reasons, I stand out like a sore thumb. Frankly, the NSA is not in my threat model. If it were, I wouldn't run servers with network services provided by programs written in C. In this article, I assume that the reader is dealing with people who have suddenly decided that the NSA is part of the threat model and that the reader needs data points to apply in a re-education process.

Traffic Analysis

A number of actions have driven folks to look for more privacy, but the core of the movement lies in that word, "privacy," and the NSA's wholesale gathering of traffic analysis data, of everyone everywhere always. PGP can help you with everything after that initial traffic analysis gathering. Traffic analysis is all about knowing who is talking to whom, and when. PGP, as an object-level privacy wrapper, not only does not hide that, it actually embeds the keys of the recipients into the message. This is optional, but almost always done, because it makes life easier for the recipients when there is information in the wrapper about which keys were used as part of encryption. These are the recipient keys that are provably tied to a given identity, if you've gone so far as to arrange for trust verification.

Unfortunately, most mail clients with PGP integration do a rather poor job of managing Bcc recipients; too often, the keyIDs of all the recipients are listed in the wrapper. If you want to send encrypted email and use Bcc, do some testing first before trusting the integration. Ideally, the Bcc'd copies will be sent as independent SMTP transactions.

If you're trying to avoid traffic analysis while still using email, then I suggest that you hide in the anonymity of crowds. That means using a mail service that provides mail for many people, not just you. If the mail service you choose uses SMTP/TLS for MX delivery, then all that an eavesdropper knows is that someone in domain A sent a message to someone in domain B. You can get a lot of protection, if you trust the mail service provider to provide privacy up until an individual legal warrant is served if both users are in the same domain, and there are enough users that the timing of the connections won't reveal anything, and neither will the sizes of data transferred. If domains A and B are large and use TLS between each other, you've doubled the number of service operators to be trusted but are still protected from traffic analysis because of the sheer volume of data continuously being exchanged between the two.

> ***Naming***
> OpenPGP is the technical name for the standards, GnuPG is a common implementation of that, as is pgp(1) from the company PGP, but most commonly the systems are simply called PGP, the name of Phil Zimmermann's original implementation.

For mail service protection against traffic analysis, the bigger the provider, the better. If your provider offers SMTP and IMAP access, you can still use PGP to protect the content instead of having to trust the provider. Of course, if most people are using Webmail interfaces, then you risk standing out.

Using PGP Safely

With that out of the way, there are issues to understand around using PGP safely. After key selection and key sizes (just use RSA with 3072 or more bits; 4096 is the practical upper bound [1]), it all comes down to key identity, knowing which keys are correct and how you retrieve the keys to use.

PGP uses the Web of Trust model, with each client making its own trust decisions. Each PGP key is a self-contained certificate, and implicitly an always-open Certificate Signing Request; the "key" passed around is a bundle of collected signatures each made by various other people's keys. Anyone can sign anyone else's keys. Be aware that here is some confusing terminology: a PGP "key" contains a cryptographic public "key" and attestations of identity, which various people will certify.

As an example, let's work with Alice, Bob, Charlotte, and Derek. Alice meets Bob, exchanges enough information with him about his key to decide that the key she has for him really is his public key. Perhaps Bob has his key fingerprint on his business card. Bob similarly verifies Charlotte's key, and Charlotte does the same with Derek. Now, if Alice can trust Bob to do a good job, and Charlotte to do a good job, despite never having met Charlotte, then she might be able to trust that she has the "right" self-contained certificate matching an identity for Derek to a public key. She might have trust in the chain.

Fundamentally, if Bob gives me his complete PGP key, I trust that the cryptographic public key contained therein is his. I might not yet trust the email address claimed in the key, or that Bob's name really is Bob, but I'll trust that the public key material presented is Bob's. Issuing a public signature is about verifying who Bob is and whether he really does own the email addresses that he claims to own.

This attestation of identity doesn't scale well, because People Are Lazy. We all know people are lazy. Anyone who has tried to get PGP adopted more widely will have dealt with folks who will sign any key, and publish that signature, not caring that they've made a public attestation of identity, saying "trust me on this," without bothering to do any checking to back that attestation. There's a reason that our society has the concept of public notaries: a possibly unfair subset of the population whom we might choose to trust to bother doing checking. And heck, we might even trust the people choosing those notaries to do an honest job.

Sure, there are some people you might trust. Sometimes just seeing the email domain is sufficient to infer something meaningful. For instance, even though I don't use Debian, I trust their training and indoctrination enough that when, during a trust database update, I'm presented with an @debian.org UID, I'll usually choose to score the key as having "marginal" trust instead of "don't know," even if I don't know who the person is.

Given that People Are Lazy, the first and biggest problem here is that the default mode for PGP clients always seems to be to create public signatures when signing someone else's key. Each key signature you make can either be "exportable," that is, public, or "local," used purely for personal convenience. The distinction is purely a Boolean in the PGP data structure of the signature, used as a hint that the signature should not normally be exported for use by others. I'll posit that most users never think through the issues enough to develop a viable mental model of the tools and concepts they're dealing with, so the default should probably be to make local signatures, with a --tell-others-to-trust-me-on-this flag to create an "exportable" signature. That simple act might encourage others to gain enough understanding to make the Web of Trust look less like a web woven by a spider on meth. Fewer signatures might appear to hurt scaling, but they'd be higher quality signatures by default.

If you want to convey more information to others about the verification you have done, PGP lets a key signature include a policy URL to provide a pointer to a description of the sorts of verification done. With GnuPG, you can use the --cert-policy-url option to set this.

Key Distribution

Who are you trusting, with what information, when you fetch a PGP public key? Are you using the public keyservers? I run one of those—I have patches in the codebase—and I think they're useful enough that I'll continue to do this as a public service for as long as I think it tenable. But public keyservers are also filled with junk. Even spam. The public keyservers do no trust-path verification. They barely manage to check that a key is valid or well-formed. They should be rejecting non-exportable key signatures, but the main peering mesh of keyservers doesn't do that and keyserver developers are trying to figure out what to do about that without breaking the key set reconciliation algorithm. Presence of a key in the public keyservers means nothing.

Furthermore, when you ask a keyserver for a key, you're communicating to that keyserver who or what it is you want to communicate with; whether a human for email, or a Web server participating in Monkeysphere. This provides information for the same traffic analysis discussed earlier. The people maintaining pool definitions of public keyservers don't validate the people running the keyservers, they just validate that there's a functional keyserver that is staying up-to-date with "enough keys that it's probably current." A spider does this validation by walking the stats pages of the keyservers, figuring out what the

Keysigning Parties

The security of PGP is largely built around being able to validate identity assertions via the Web of Trust. Although anyone can validate another's key at any time, there are scaling efficiencies to organized events. You will often find in the schedule for technical conferences a BoF type session that facilitates mass cross-signings. It's called a "party," but that is surely someone's idea of a joke. You only need to attend one or perhaps two of these events to have your key end up in the "Strong Set," about which you can find more details online.

In short, all folks planning to attend let the organizer know ahead of time, with their keyIDs. The organizer prepares a keyring with those keyIDs and prints out sheets of paper with each key and fingerprint on it. Each attendee receives a copy of that at the event. During the event, some mechanism will be used to let each person see every other person's photo ID; depending upon attendee count, that might be passing cards around, or a somewhat sophisticated conga line that will eventually dissolve into chaos. But it's still not a party. Then each attendee in turn will stand up and read out their key fingerprint, from their own trusted source (not the paper copy prepared by the organizer), letting each other attendee check off the details. By the end of this, each attendee should have some confidence in a legally accepted human name attached to each keyID. Verifying the email addresses is then a matter of sending the signature of each PGP UID to the address in that UID, encrypted to the key, shifting the responsibility to the key-owner to upload the key to public keyservers. The two main tools to automate this are "caff" and "pius" [2]. This means that the extent of the email verification is "someone with access to the mailbox also had access to the private key and was happy to affirm the association."

peering mesh is, and determining which keyservers can be used to get current keys. The spider then updates DNS to update the records returned for various pools, including geographic pools.

If intelligence agencies aren't running public keyservers under hostnames designed to sound cypherpunkish, to get a percentage of traffic analysis about who wants to talk securely with whom, then they're slipping. They could skip this step, as the communication between keyservers is an old protocol using HTTP with a fixed pattern of URL construction, no matter which host is chosen. This protocol is called HKP, which stands for Horowitz or HTTP Keyserver Protocol, depending upon whom you ask. The traffic is almost always unencrypted. A well-placed traffic tap for data flowing to and from the default HKP TCP port, 11371, is probably very informative.

The non-keyserver approaches usually involve tools such as finger, also unencrypted.

If you want people in your organization to have some privacy in whom they're communicating securely with (end-to-end), consider running a local "SKS" keyserver: you'd currently need to also provide this as a public service as an inherent part of how you exchange traffic with peers, but the front-end HTTP proxy you put up can also offer HTTPS (HKPS) communication on a known hostname, so that there's an identity your software clients can validate, using the PKIX, which is an entirely different can of worms. If you have an internal certificate authority, and manage internal software deployments enough to control default GnuPG configurations, you can at least ensure that only your CA is trusted for keyserver host identities.

There is little HKPS in the public PGP keyserver web because most client communication is via pool hostnames, and getting PKIX signatures for pool services run by unaffiliated independent groups certainly should be impossible. And even if you had that, it would be incentive for some well-funded groups to offer keyservers that happen to forward logs of retrievals to some (perhaps local) acronym agency.

There are no better solutions for OpenPGP on the horizon if you're concerned about traffic analysis. Various systems that put keys into DNSSEC-secured DNS can provide for confidence that you have the right key, but certainly don't protect against a network traffic filter for DNS traffic concerning the CERT or OPENPGPKEY RR-types.

Anything using well-known URLs or services in the recipient's domain will leak some traffic data when you retrieve the key; in the best-case scenario, where the link is encrypted, it's obscured to the domain level. In the worst-case, you've just signaled to the world that now would be a good time to compromise your client machine.

Good luck helping your paranoid users thump hard back into reality.

References

[1] For more guidance on choosing key lengths, see http://www.keylength.com/en/3/, the "Asymmetric" table column. This site provides guidance from several sources, so you can pick and choose depending upon whom you trust most for advice.

[2] Tools to help with keysigning parties: pius, http://phildev.net/pius/; caff, http://pgp-tools.alioth.debian.org/.

usenix THE ADVANCED COMPUTING SYSTEMS ASSOCIATION

11th USENIX Symposium on Operating Systems Design and Implementation (OSDI '14)

Sponsored by USENIX, the Advanced Computing Systems Association

October 6–8, 2014, Broomfield, CO

Important Dates

Abstract registration due: *Thursday, April 24, 2014, 9:00 p.m. PDT*

Complete paper submissions due: *Thursday, May 1, 2014, 9:00 p.m. PDT*

Notification to authors: *Thursday, July 17, 2014*

Final papers due: *Wednesday, September 10, 2014*

Symposium Organizers

Program Co-Chairs

Jason Flinn, *University of Michigan*
Hank Levy, *University of Washington*

Program Committee

Atul Adya, *Google*
Lorenzo Alvisi, *University of Texas, Austin*
Dave Andersen, *Carnegie Mellon University*
Remzi Arpaci-Dusseau, *University of Wisconsin-Madison*
Mihai Budiu, *Microsoft Research*
George Candea, *EPFL*
Peter Chen, *University of Michigan*
Allen Clement, *Max Planck Institute for Software Systems*
Landon Cox, *Duke University*
Nick Feamster, *Georgia Tech*
Bryan Ford, *Yale University*
Roxana Gaembasu, *Columbia University*
Steve Gribble, *University of Washington and Google*
Gernot Heiser, *University of New South Wales*
Frans Kaashoek, *Massachusetts Institute of Technology*
Kathryn McKinley, *Microsoft Research*
Ed Nightingale, *Microsoft*
Timothy Roscoe, *ETH Zurich*
Emin Gün Sirer, *Cornell University*
Doug Terry, *Microsoft Research*
Geoff Voelker, *University of California, San Diego*
Andrew Warfield, *University of British Columbia*
Junfeng Yang, *Columbia University*
Yuanyuan Zhou, *University of California, San Diego*
Willy Zwaenepoel, *EPFL*

External Review Committee

Emery Berger, *University of Massachusetts*
Luis Ceze, *University of Washington*
Angela Demke Brown, *University of Toronto*
Greg Ganger, *Carnegie Mellon University*
Joseph Gonzalez, *University of California, Berkeley*
Andreas Haeberlen, *University of Pennsylvania*
Galen Hunt, *Microsoft Research*
Sam King, *Adrenaline Mobility and University of Illinois*
Eddie Kohler, *Harvard University*
Ramakrishna Kotla, *Microsoft Research*
Jinyang Li, *New York University*
Wyatt Lloyd, *Facebook and University of Southern California*
Shan Lu, *University of Wisconsin*
Jeff Mogul, *Google*
Satish Narayanasamy, *University of Michigan*
Jason Nieh, *Columbia University*
Vivek Pai, *Princeton University*
Rodrigo Rodrigues, *Universidade Nova de Lisboa*
Bianca Schroeder, *University of Toronto*
Mike Swift, *University of Wisconsin*
Kaushik Veeraraghavan, *Facebook*
Hakeem Weatherspoon, *Cornell University*
Matt Welsh, *Google*
John Wilkes, *Google*
Emmett Witchel, *University of Texas*
Ding Yuan, *University of Toronto*
Nickolai Zeldovich, *MIT CSAIL*
Feng Zhao, *Microsoft Research*

Steering Committee

Remzi Arpaci-Dusseau, *University of Wisconsin-Madison*
Brad Chen, *Google*
Casey Henderson, *USENIX*
Brian Noble, *University of Michigan*
Margo Seltzer, *Harvard School of Engineering and Applied Sciences and Oracle*
Chandu Thekkath, *Microsoft Research Silicon Valley*
Amin Vahdat, *Google and University of California, San Diego*

Overview

The 11th USENIX Symposium on Operating Systems Design and Implementation seeks to present innovative, exciting research in computer systems. OSDI brings together professionals from academic and industrial backgrounds in what has become a premier forum for discussing the design, implementation, and implications of systems software. The OSDI Symposium emphasizes innovative research as well as quantified or insightful experiences in systems design and implementation. OSDI takes a broad view of the systems area and solicits contributions from many fields of systems practice, including, but not limited to, operating systems, file and storage systems, distributed systems, cloud computing, mobile systems, secure and reliable systems, embedded systems, virtualization, networking as it relates to operating systems, management and troubleshooting of complex systems. We also welcome work that explores the interface to related areas such as computer architecture, networking, programming languages, and databases. We particularly encourage contributions containing highly original ideas, new approaches, and/or groundbreaking results.

Submitting a Paper

Submissions will be judged on originality, significance, interest, clarity, relevance, and correctness. All accepted papers will be shepherded through an editorial review process by a member of the program committee.

A good paper will:

- Motivate a significant problem
- Propose an interesting, compelling solution
- Demonstrate the practicality and benefits of the solution
- Draw appropriate conclusions
- Clearly describe the paper's contributions
- Clearly articulate the advances beyond previous work

All papers will be available online to registered attendees before the conference. If your accepted paper should not be published prior to the event, please notify production@usenix.org. The papers will be available online to everyone beginning on the first day of the conference, October 6, 2014.

Papers accompanied by nondisclosure agreement forms will not be considered. All submissions will be treated as confidential prior to publication on the USENIX OSDI '14 Web site; rejected submissions will be permanently treated as confidential.

Simultaneous submission of the same work to multiple venues, submission of previously published work, or plagiarism constitutes dishonesty or fraud. USENIX, like other scientific and technical conferences and journals, prohibits these practices and may, on the recommendation of a program chair, take action against authors who have committed them. See the USENIX Conference Submissions Policy at www.usenix.org/conferences/submissions-policy for details.

Prior workshop publication does not preclude publishing a related paper in OSDI. Authors should email the program co-chairs, osdi14chairs@usenix.org, a copy of the related workshop paper and a short explanation of the new material in the conference paper beyond that published in the workshop version.

Questions? Contact your program co-chairs, osdi14chairs@usenix.org, or the USENIX office, submissionspolicy@usenix.org.

By submitting a paper, you agree that at least one of the authors will attend the conference to present it. If the conference registration fee will pose a hardship for the presenter of the accepted paper, please contact conference@usenix.org.

If your paper is accepted and you need an invitation letter to apply for a visa to attend the conference, please contact conference@usenix.org as soon as possible. (Visa applications can take at least 30 working days to process.) Please identify yourself as a presenter and include your mailing address in your email.

Deadline and Submission Instructions

Authors are required to register abstracts by 9:00 p.m. PDT on April 24, 2014, and to submit full papers by 9:00 p.m. PDT on May 1, 2014. These are hard deadlines. No extensions will be given. Submitted papers must be no longer than 12 single-spaced 8.5" x 11" pages, including figures and tables, plus as many pages as needed for references, using 10-point type on 12-point (single-spaced) leading, two-column format, Times Roman or a similar font, within a text block 6.5" wide x 9" deep. Final papers may gain two pages, for a total of 14 pages. Papers not meeting these criteria will be rejected without review, and no deadline extensions will be granted for reformatting. Pages should be numbered, and figures and tables should be legible in black and white, without requiring magnification. Papers so short as to be considered "extended abstracts" will not receive full consideration.

Papers must be in PDF format and must be submitted via the Web submission form on the Call for Papers Web site, www.usenix.org/osdi14/cfp.

Blind reviewing of full papers will be done by the program committee and external review committee, with limited use of outside referees. Authors must make a good faith effort to anonymize their submissions, and they should not identify themselves either explicitly or by implication (e.g., through the references or acknowledgments). Submissions violating the detailed formatting and anonymization rules will not be considered for publication.

Authors are encouraged to contact the program co-chairs, osdi14chairs@usenix.org, if needed to relate their OSDI submissions to relevant submissions of their own that are simultaneously under review or awaiting publication at other venues. The program co-chairs will use this information at their discretion to preserve the anonymity of the review process without jeopardizing the outcome of the current OSDI submission.

For more details on the submission process, and for templates to use with LaTeX, Word, etc., authors should consult the detailed submission requirements linked from the Call for Papers Web site.

Jay Lepreau Award for the Best Paper

The program committee will, at its discretion, determine which paper(s) should receive the Jay Lepreau Award for the Best Paper.

Poster Session

We plan to hold a poster session in conjunction with a social event at the Symposium. Details on submitting posters for review will be available on the Web site by August 2014.

Registration Materials

Complete program and registration information will be available in August 2014 on the conference Web site.