Digital Security HOWTO: Protect your Data, Communications, and Activities, & Painlessly Integrate Teaching Simple Security Into Classes

Jonathan A. Poritz

Department of Mathematics and Physics
and Center for Teaching and Learning
Colorado State University – Pueblo
2200 Bonforte Blvd.
Pueblo, CO 81001-4901

jonathan@poritz.net
http://poritz.net/jonathan

Domains 2017: Indie EdTech and Other Curiosities
Hosted by University of Oklahoma, 5-6 June 2017

These slides can be found at https://poritz.net/j/share/dom17

Background, Pt. 1: I am *not* a Luddite

I've been programming since I was in middle school [programming was not a good way "to learn the value of money," as my parents hoped: I was paid too much].

I've owned `poritz.net` for 12 years (= 84 in dog years and 1100 years in Internet years*).

Like many of the people in this room [ask my colleague and co-author Jonathan Rees what he does with technology in his history courses!], I teach in ways that would have been impossible just a few years ago, because of bleeding-edge technologies.

Philosophically, I think information technology is one of a small handful of epochal transformations in the relationship of *homo sapiens* to the world, *viz.*:

- tool use – expends the reach and power of our bodies ... although chimpanzees, crows, dolphins, ants, and other animals also make tools;
- language – allows the movement of thoughts from one head to another, and hence to (oral) culture, sophisticated cooperation, *etc.* ... although bees, some individual primates, and likely cetaceans also have language;
- writing – allows the movement of ideas across great distances in space and time, and the extension of the size of "working memory" so individuals can have more complex thoughts [maybe allowing the invention of mathematics?];
- computation – allows *thinking to happen outside of human heads!*

Whatever their problems (as we are about to discuss), computers and the Internet do make me smarter, do [sometimes] provide opportunities for beauty and activities in the pursuit of justice, structure much of the world in which many of our students will work and live and play, and therefore efforts such as DoOO to build good digital citizens are *vital*.

*To convert a number of calendar years into Internet years, express the number in base 2 but then think of that sequence of digits as expressing a number in base 10.

Background, Pt. 2: And yet, *L'enfer, c'est l'Internet.*[*]

The Internet was conceived during the MAD old days of the cold war, and had its formative childhood and early adolescence during the post-Reagan/Thatcher rise of neoliberalism.*[With an origin story like that, it's kind of amazing that it's doing so well!]*

Some circles of Internet Hell:

1. it is a neoliberal dystopia – what is commonly called *surveillance capitalism* is probably better named *vampire capitalism*;
2. "fake news" – [largely a cultural issue? see `nytimes.com/2017/05/08/world/europe/macron-hacking-attack-france.html`];
3. trolling/racism/misogyny – [the fault of neoliberalism?];
4. surveillance and censorship;
5. computer crime

Instruction in digital citizenship must include these topics.

[*] with apologies to Jean-Paul Sartre.

Firing back: Against the tyranny of the market*

Our strongest weapon against points **4** [surveillance/censorship] and probably **1** [neoliberalism] is *FLOSS*.

Richard Stallman's term *free software* is preferable to the much inferior *open-source software*, for many of the reasons `rms` gives in his works **but also** because the word *free* draws attention to a connection with

- *artistic freedom*
- *freedom of speech/thought/conscience/etc.*
- *academic freedom*
- *liberal-arts education* [the origin of which was "education in the arts of freedom/being a free person"]

FLOSS = "Free/Libre Open-Source Software" is a bit of a cop-out term, stepping away from `rms`'s Old Testament prophet-like thunderings [God-given though they may be] in the guise of simply compensating for the poverty of words in English for particular kinds of freedom. But it is better than plain "open-source software."

[See discussion in *Education is Not an App*, book with Jonathan Rees.]

*with apologies to Pierre Bourdieu

"No Smoking"

Watching people at OER conferences, and other conferences about Internet-related topics with the word *open* in their titles, walking around talking on their `iPhones` and typing away on their `MacBooks` reminds me of a time I was going up a stairwell in a hospital in Rome and came across a half-dozen doctors and nurses furiously puffing away on their cigarettes under a sign the said *Vietato Fumare*, outside the entrance to the neonatal ICU.

But that's not what this talk is about.

It's about what we can teach our students about, and put into practice ourselves [on our DoOO sites or other Internet places where we have any agency], about **security**, in response to the infernal circles **5** [computer crime] and **4** [surveillance/censorship], above.

To discuss this topic, we must understand some basic *cryptology*, because without understanding, everything we do is empty pointing-and-clicking, full of sound and fury, signifying nothing.

Please imagine scary mood music.

Crypto tends to intimidate users, because they hear that small mistakes can be fatal. But that is literally true of hundreds of decisions we make every minute while driving on the highway, yet we expect just about anyone of a certain age, even without fancy academic credentials, to be able to get a driver's license.

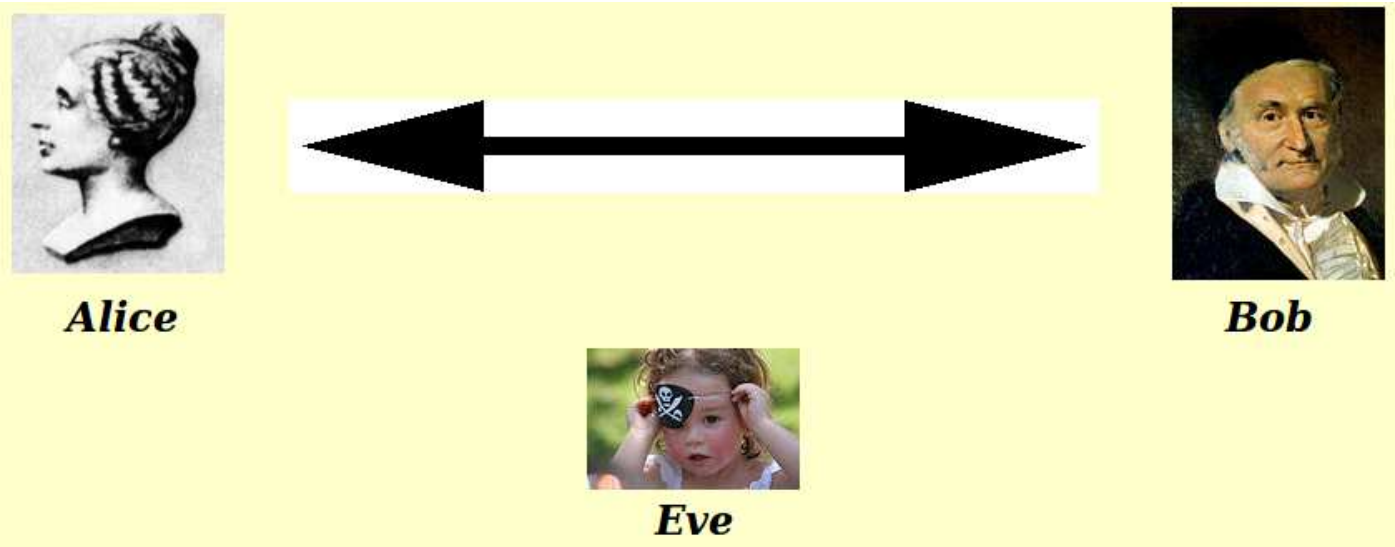
The other problem with crypto is that it is fairly *mathy*. Which is a bit like the Kaiser saying "Too many notes, my dear Mozart" – it's a feature, dahling.

There will be a short quiz at the end of this session on this material. The good news is that it will be "open book/notes/Internet," and there are great resources on the 'net. For example:

- Chapter 4 of my open textbook **Yet Another Introductory Number Theory Textbook** [**YAINTT**], which can be found at `poritz.net/jonathan/share/yaintt.pdf` [although, being a math textbook, this is rather unabashedly *mathy* – even though that Chapter has a lot of history and terminology.]; or
- Ed Felton's [Princeton professor of computer science and Deputy Chief Technology Officer under Obama] **Nuts and Bolts of Encryption: A Primer for Policymakers**, found at `https://www.cs.princeton.edu/~felten/encryption_primer.pdf`, which is not *mathy* at all.

Our protagonists, and an adversary

Most works of cryptology speak of two star-crossed lovers, *Alice* and *Bob*, who attempt to keep the guttering candle of their love alight, though distance separates them and their communications are being monitored by the evil *Eve*.



[Extra credit if you can name the two famous mathematicians who acted as models for these pictures of Alice and Bob.]

Why is Eve so powerful?

It's important to realize that in many – maybe **most** – situations, it is entirely appropriate to assume that Eve can see all the communication between Alice and Bob while it is in transit. All of the channels you are used to using suffer from this:

- A cell phone is basically a *walkie-talkie with infrastructure* [the infrastructure being all those cell towers all over the place]. Anyone with a radio receiver of the right type who is within the footprint of the same tower can hear the entire exchange. [**Stingrays!**]
- Satellite phones are much worse: the footprint is the size of a continent, often.
- Anything you do on the Internet is essentially public. [Ask your millennial students how the basic Internet Protocol works – you will be surprised/horrified at what they say.]

Here is a diagram with some basic terminology:

Alice	on public network	Bob
plaintext/cleartext message m **encrypts** m to c transmits c	$\rightarrowtail$ **ciphertext** c $\rightarrowtail$	receives c **decrypts** c to recover plaintext m

Kerckhoffs *vs* security-by-obscurity

In the design of the encryption and decryption algorithms, we follow something cryptologists call **Kerckhoffs' Principle** [named after *Auguste Kerckhoffs* a professor of languages at the École des Hautes Études Commerciales in Paris in the late 19^{th} century who wrote influential papers on cryptology]. According to this Principle, one always publishes the details of one's cryptographic algorithms.

It may seem ridiculous to publish the algorithm used to protect your data, but we do this because humans have a nearly infinite capacity for self-deception. As a consequence, we are always thinking we have invented the best cryptographic algorithm, a perpetual motion machine, the way to square the circle and trisect the angle ... when another set of eyes, looking over our work independently, would immediately see flaws. This is nothing other than the famous idea of peer review, of course!

[The alternative to putting your proposed cryptographic algorithms out in the world for peer review is called by cryptologists with enormous disdain *security by obscurity*. Experience has shown that it is no security at all.]

Keys [for symmetric cryptosystems]

If we are to publish our encryption and decryption algorithms, the security must lie in some other secret. This is an additional piece of information called the **key**, which is input into those algorithms, as follows:

Alice $\longleftrightarrow$	private communication of shared key $k \in \mathcal{K}$ $\longleftrightarrow$	Bob
	on public network	
message $m_A \in \mathcal{M}$ compute $c_A = e_k(m_A)$ transmit c_A	$\rightarrowtail$ ciphertext c_A $\rightarrowtail$	receive c_A compute $m_A = d_k(c_A)$
receive c_B compute $m_B = d_k(c_B)$	$\leftarrowtail$ ciphertext c_B $\leftarrowtail$	message $m_B \in \mathcal{M}$ compute $c_B = e_k(m_B)$ transmit c_B
etc....		

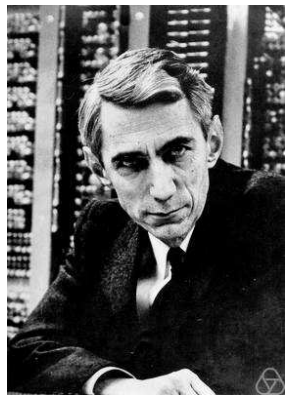
Notes on symmetric cryptosystems

The above is called **symmetric** (or **private-** or **secret-key**) **cryptography**. We shall see an alternative in a few minutes.

Notes:

- Both the encryption e_k and decryption d_k use the same key k, which must be shared in some private, pre-lapsarian moment. The keyspace $\mathcal{K}$ must be large, otherwise Eve can just try all keys and see which works [which is called a **brute-force attack**].
- Symmetric cryptosystems are fast you can run a video stream through one without noticing it on a consumer-grade PC.
- The design of symmetric cryptosystems is something of a black art. There is little general theory on the attack or defense side, and the algorithms tend just to be along the lines of *scramble the bits a lot.*
- Some examples:
  - The *Syctale* – ancient Greece
  - The *Caesar cipher* – actually used by Julius Caesar. [addition mod 26...]
  - The *Vigenère Cipher* – thought to be unbreakable for centuries. Easy to break today.
  - The *one-time pad* – completely **unbreakable**; hard to use in practice (but see Leo Marks' **Between Silk and Cyanide: A Code Maker's War 1941-45**)
  - The *Enigma machine* – a German military coding device from WWII.
  - Modern block ciphers like *DES, triple-DES, AES, etc.*

Symmetric Encryption of Data Standing Still



Actually, the communication channel could be from *past you* to *future you; i.e.*, we're just encrypting stored data. This is a good idea. Claude Shannon had this idea (to use crypto from communications also in this context), and many other important ones.

Examples:

- Please, please use full-disk encryption. Model this good behavior for your students, as well.
- Demonstration of using **GnuPG** for encryption:
  `gpg --output <file.gpg> --cipher-algo AES256 --symmetric <file>`
  and decryption
  `gpg <file.gpg>`
  Look at the file with
  `hexdump -C <file.gpg>`

I don't know how to use GnuPG in a GUI, although I presume there is a way. This should not be a problem – we tend to be very verbal people.

GUIs are like bashing orcs with a magic sword, while working on the command line is like speaking the words of a spell which causes peace quietly to arrive in a troubled land.

Asymmetric cryptosystems

If Alice and Bob want to be able to communicate securely without ever having met to exchange the symmetric key, they can instead use **asymmetric** (or **public-key**) **cryptography**:

Alice	on public network	Bob
		pick $k_d \in \mathcal{K}_d$ compute $k_e = \mathcal{E}(k_d) \in \mathcal{K}_e$
download k_e	$\longleftarrow$ public key k_e $\longleftarrow$	publish k_e
message $m \in \mathcal{M}$ compute $c = e_{k_e}(m)$		
transmit c	$\longmapsto$ ciphertext c $\longmapsto$	receive c
		compute $m = d_{k_d}(c)$

That this is possible at all is very cool. There are a few ways we do it, now, including **RSA** (named after Ron Rivest, Adi Shamir, and Leonard Adelman, who published this idea in 1977) and **elliptic curves** (which are more efficient but less commonly used, since their mathematics is significantly harder to chew that what is behind RSA).

All asymmetric crypto relies upon a mathematical function which is easy to compute in one direction but difficult to invert. For RSA, this is essentially multiplication forward [easy], but factoring backwards [hard]. For other asymmetric algorithms, there are other of these *one-way functions.*

The "Man-in-the-middle attack"

A significant issue with asymmetric cryptosystems is a **Public-key Infrastructure** [**PKI**], because of the dreaded **man-in-the-middle attack**:

Alice	Eve	Bob
	intercept $k_e^B \leftarrowtail$	generate keys: public k_e^B, private k_d^B publish k_e^B
download k_e^E	generate keys: public k_e^E, private k_d^E $\leftarrowtail k_e^E$, spoof origin	
message $m \in \mathcal{M}$ compute $c_A = e_{k_e^E}(m)$ transmit c_A	$\rightarrowtail c_A$, intercept	
	extract the cleartext $m = d_{k_d^E}(c_A)$ change to m' if desired compute $c_E = e_{k_e^B}(m')$ spoof origin $c_E \rightarrowtail$	receive c_E
		read the message $m' = d_{k_d^B}(c_E)$

Digital Signatures

Therefore, we need to be sure that the public keys we use really do belong to the people who we think they do. We do this either by getting the key from someone in person – but that kind of ruins the whole idea of asymmetric crypto! – or we get a key in some way that we are sure of its provenance. One way to be sure would be to have a **digital signature** on the public key, signed by someone whom we trust. Signatures work like this:

Larry	on public network	Bob
		pick $k_d \in \mathcal{K}_d$ compute $k_e = \mathcal{E}(k_d)$
download k_e	$\longleftarrow$ public key k_e $\longleftarrow$	publish k_e
		message $m \in \mathcal{M}$ compute $s = d_{k_d}(m)$
receive (m, s)	$\longleftarrow$ signed message (m, s) $\longleftarrow$	transmit (m, s)
if $e_{k_e}(s) = m$ ACCEPT otherwise, REJECT		

Certificate Authorities or key-signing parties

Signatures on public keys are called **certificates**, and you have to trust **their** public key to use them, or else check a signature on the certificate signer's key, and on recursively as far as necessary. In the end, there are certain **Certificate Authorities** whose keys are baked into many common devices, so that establishes a root of trust. This can be very good, in building reliable trust in software, or bad if it bakes into a particular OS or service a requirement to participate in some closed software ecosystem. [This is Apple's business model with the `iPhone`, for example.]

Another, less formal, approach is for individuals to sign each other's keys, when they know each other personally, until gradually there is a large **web of trust**. The fun way to do this is to throw a key-signing party where people who know each other bring laptops and sign each other's keys.

Practical Successes with Asymmetric Cryptosystems: Mailvelope

Long ago, in an Internet far, far away ... the Crypto Wars were fought.

They were started then (1991) by Phil Zimmerman, who released *Pretty Good Privacy*. The Feds sued, he was defended by Eben Moglen (founder of the Software Freedom Law Center). Eventually, the good guys won.

Recently, the *OpenPGP* standard (`https://tools.ietf.org/html/rfc4880`) was implemented in *javascript*, as **OpenPGP.js**. Let's install and use a FLOSS Firefox and Chrome extension which does public-key crypto for common webmail clients: **Mailvelope**, `http://mailvelope.com/`.

The third-party doctrine in 4^{th} Amendment law suggests we should keep only the encrypted versions on the webmail provider's servers. **Mailvelope** does this. It also keeps track of your keys ... protected by a password and the security of your machine. [So there is not much point in using this under *Windoze*, because its security is so spectacularly weak.]

A public key you can use to try this is at `poritz.net/jonathan/share/ofsamos.asc` . Feel free to send email encrypted under this key to `ofsamos@gmail.com`; Pythagoras [of Samos] will reply.

`https`, Let's Encrypt, HTTPS Everywhere

The *http* protocol running over SSL (so, with public-key based security) is called `https`, and shows up as that little lock in the URL bar (or corner) of your browser window. It is quite safe: feel free to go to private sites, login to your webmail, *etc.*, when using it, even on public wifi.

To enable `https` on a site, the site needs an SSL certificate to prove its public key is valid in an man-in-the-middle-proof way. Such certificates used to be very expensive, but the **Electronic Frontier Foundation** [**EFF**] (`eff.org`) has a great project called **Let's Encrypt** which makes it free and painless. Not every web hosting service allows this! [**Reclaim** does, of course.]

If you enable SSL (with a certificate) on your site, it will be more likely to be accessible when embedded inside other, security-conscious services like LMSs. Also, search engines will give preferences to your site over otherwise similarly ranked sites for the same keywords.

The EFF has a Firefox and Chrome plugin **HTTPS Everywhere** which forces your browser – so, on the *client* side – the `https` connection to a site, if it is supported – install and use this!

Final Thoughts [RIP?]

- How are people in the academic world comfortable using nonfree software?
- Things you can do to model good security practice:
  - Use a good password. [Your current password is very weak, almost certainly, even if it is very hard for you to remember.]
  - Encrypt all of your drives.
  - Encrypt your email.
  - Get an SSL certificate for your site.
  - Use **HPPS Everywhere**, **Privacy Badger**,
- Ways to get your posse [students, faculty, staff] started:
  - Make a rule that you will only accept [certain kinds of] email from students if it is encrypted?
  - Make installing SSL (from **Let's Encrypt**) a standard [required] practice in your class or institution or DoOO operation.
- Think and talk about security **all the damn time**.
  - Don't admit you're afraid of it.
  - There are **many**, **fantastic** books/HOWTOs/manpages/Khan Academy Videos/*etc.*
  - Ask a cryptologist when you're concerned. Most of us are very nice people.

Because the Internet sucks.*

*But it's pretty cool, too.