Public Key Cryptography

PGP, OpenSSL and friends!

Joseph Hallett

January 22, 2025

University of BRISTOL

I want to send you a secret!

How are we going to do it?

I could encrypt it with a password...

For how to encrypt something see a cryptographer...
But lets just pretend we have a magic cryptography box

- We give it a password and some data...
- We get out encrypted data

But there's a problem...?

How do I give you the password?

In security we have this idea of a threat model where we detail everything we are concerned might spoil our fun

- Suppose we're concerned with an all powerful network administrator who can see everything we say...
- We would need to be able to send the password without sending the password
- We need a way of transmitting a password using only public information

Hmm... this seems hard...

This seems hard lets try a different problem!

I want to check you are who you say you are!

How are we going to do it?

I could check your ID...

But how do I know it’s your ID?

- How do I know it isn’t fake?

How do I trust who issued it?

- How do I know an evil admin hasn’t printed their own new ID?

Public Key Cryptography

The solution to both these problems is something called public key cryptography
The more realistic version of this problem is:

- Hey is this website really `amazon.com`?
- Here's my bank details so I can buy a cute light up duck! Hope no one steals them!

You use this daily

- Connect to a website using HTTPS?
- Verifying your software hasn't been tampered with?
- Authenticating with a server?

You're already using public key cryptography to do all this

- TLS certificates with OpenSSL
- Digital signatures with GnuPG
- SSH keys with OpenSSH

So what is public key cryptography?

It's a maths problem! We're gonna keep this nicely high level, but ANY cryptography unit or book will cover the details.

- There is more than one version of it (RSA, ElGammal, DSA, Eliptic Curve Cryptography...)
- They all have trade offs

But basically we need a maths problem that is:

- Hard to solve
- Easy to solve if you know the secret
- Doesn't give away how you solved it if you do

The public key is a statement of the problem

- Anyone can see this

The secret key is the trick you need to know to solve the problem

- No one can see this

For example

RSA is a classic public key cryptosystem based around the belief that it is hard to work out the prime factors of a number

- Discovered by Ron Rivest, Adi Shamir and Leonard Adleman in 1977
- Discovered secretly by Clifford Cocks working at GCHQ in 1973
  - Founded the Heilbronn Institute at Bristol Uni (based in the Fry Building up the road)

Encryption and decryption is done by solving:

$$(\mathsf{message}^{e})^{d} \equiv \mathsf{message} \mod n$$

Public key (e, n)

Private key d

Where:

- p and q are prime numbers
- $n = p \cdot q$
- $\lambda(n) = \mathrm{lcm}(p-1, q-1)$
- e is a number such that $1 < e < \lambda(n)$ and $\gcd(e, \lambda(n)) = 1$
- $d \equiv e^{-1} \mod \lambda(n)$

Aside

Please never actually implement this:

- The gotchas involved in doing so without leading to hilarious bugs are incredibly subtle and tricky
- Friends don't let friends write cryptography libraries

If anyone says they can do it safely:

- Be incredibly suspicious
- Especially if they have a PhD in security
- Even if they have a PhD in cryptography

RSA isn't safe if you have a quantum computer… probably

- Smaller keys are broken now, larger keys broken eventually
- We have more modern cryptosystems but this is already way too much maths

So how do we use this thing?

- Alice has the public and secret key pair.
- Bob just has Alice's public key

To send a secret message to Alice

To encrypt a message using Alices's public key to set a problem that can only be solved if you know the private key

- For RSA: what was the original message if I give you message^e?
  - Easy to solve if you know d: $(\text{message}^e)^d \equiv \text{message} \mod n$
  - Hard to find a d that recovers the message if you don't

To prove that Alice sent you something

Alice creates a digital signature by using their private key to create something that recovers the original file if you know the public key

- For RSA: /Since $(\text{message}^e)^d \equiv \text{message} \mod n$ then $(\text{message}^d)^e \equiv \text{message} \mod n$
  - Heres a (message^d) that when you encrypt it with the public key will give you the original message
  - Since only Alice knows d, then Alice must have created the signature

So that's the theory...

- Everyone has two keys, one private, one public
- To encrypt something for someone: use the public key
- To prove you did something: use the private key

How do we use this in tools?

Way back in the heady days of week 1...

- Matt talked about SSH keys for logging into the labs remotely
- You created a public and private key...

So what is going on here?

- (In a very simplified way)

Can I ssh in please (with a key)?

You want to authenticate to a server as a user (`ssh user@server`)

- The server checks the user's `~/.ssh/authorized_keys`:
  - This contains the public keys for all the SSH keys which are allowed to log in as the user to this server.
- The server comes up with a challenge
  - Usually something random so you can't just replay old challenges
  - Usually something with a timestamp so the message expires if it isn't answered immediately

And then it goes something like this:

Server to User Hey if you're really this user sign this challenge with this public key...

But how do you know it's your key?

This protocol is great (we've been using it all term...) but there are a couple of issues:

- You still have to get your key onto the server (so you either need a password initially or pre-issued keys)
- If you have a shared server, your `~/.authorized_keys` file is gonna get big
- What happens if you have more than one server?
- What happens if you need to revoke someone's key
  - (a.k.a. how do we kick you out of our systems after you get fired/lose your key)

An alternative plan!

Instead of letting users upload their keys…

- Lets have a central authentication service
- You still generate your key, but instead of uploading it to the server you'll show the public key to the authority
  - They sign your key (hopefully checking you are who you say you are) and hand you back the certificate
    - Yes we're now calling the signature a certificate… isn't naming a wonderful thing?
    - It also contains some other stuff (expiry dates, who signed it, your public key…)
  - They also generate a revocation certificate to say that your key is no longer good
    - But they keep that to themselves unless they need it.
- When you go to log in, you hand over the certificate…
  - If its a valid signature of your public key by the authority…
    - And they don't have a revocation certificate from the authority disavowing the certificate
    - You're in!

This is called certificate-based authentication

- Check `man sshd_config` for how to set it up

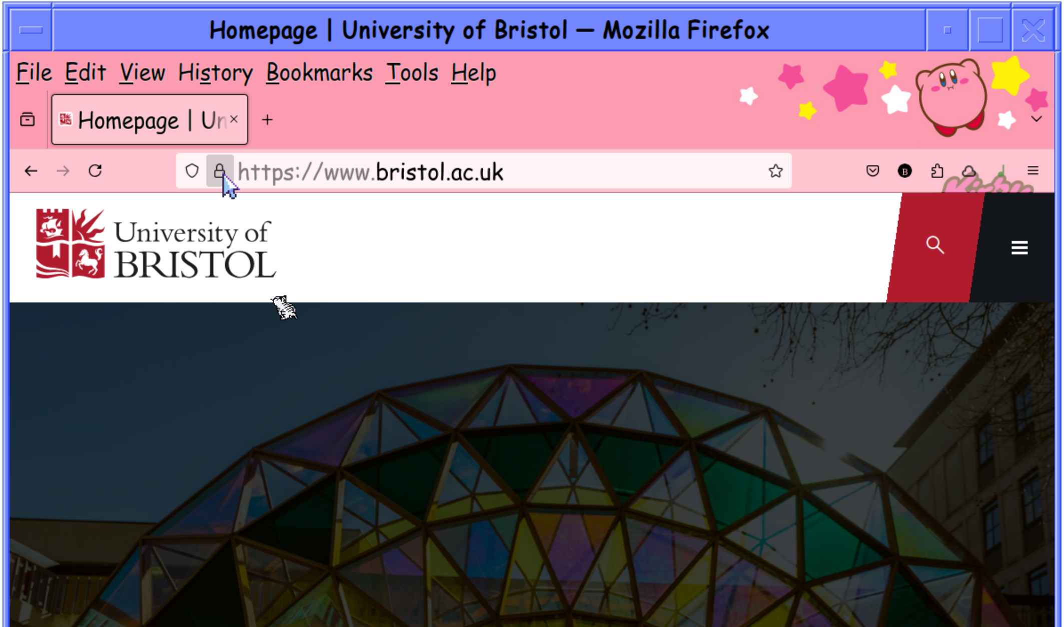
But the same basic protocol is used in a bunch of other places too

HTTPS

When you connect to a website you get a little lock appear in your URL

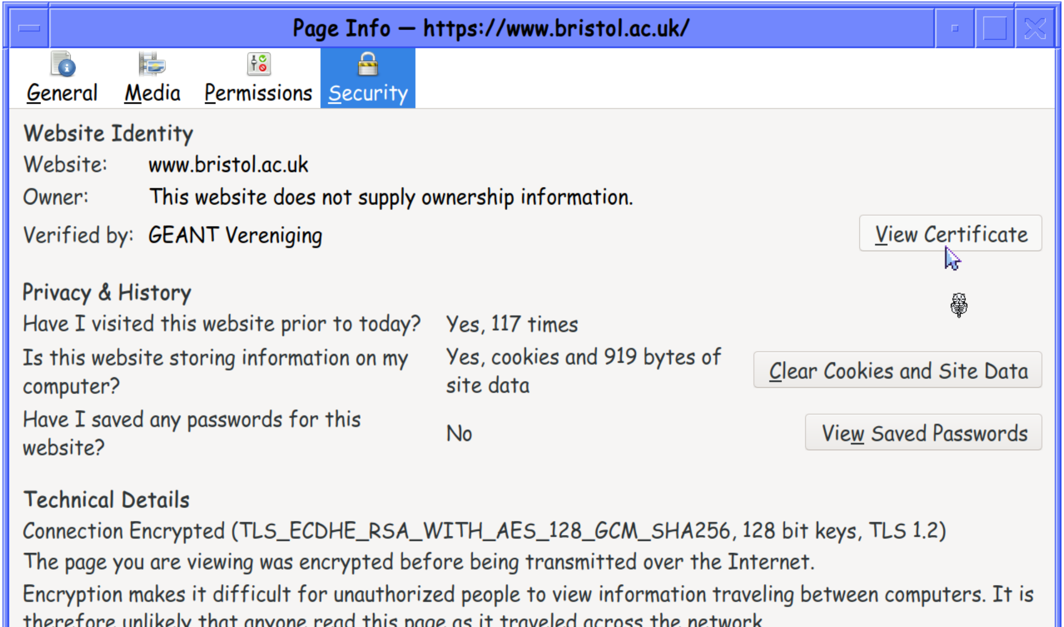
- What's all that about then?

A wild lock appeared!



Homepage | University of Bristol — Mozilla Firefox
File Edit View History Bookmarks Tools Help
Homepage | Un ×
+
https://www.bristol.ac.uk
University of
BRISTOL

A wild certificate appeared!



Website Identity

Website: www.bristol.ac.uk

Owner: This website does not supply ownership information.

Verified by: GEANT Vereniging

Privacy & History

Have I visited this website prior to today? Yes, 117 times

Is this website storing information on my computer? Yes, cookies and 919 bytes of site data

Have I saved any passwords for this website? No

Technical Details

Connection Encrypted (TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256, 128 bit keys, TLS 1.2)

The page you are viewing was encrypted before being transmitted over the Internet.

Encryption makes it difficult for unauthorized people to view information traveling between computers. It is therefore unlikely that anyone read this page as it traveled across the network.

GUI schmooie!

We could keep going with the GUI

- But this is a unit on Software Tools…

OpenSSL

- Open source organization who make cryptographic software
  - …including the `openssl` tool and library
- Named after the SSL protocol that we formerly used to securely connect to websites with
  - All versions of the SSL protocol are now broken
  - Use a recent version of TLS instead
  - Sometimes were lazy and refer to TLS as SSL
    - Don't do this. NEVER use actual SSL

OpenSSL is usable

Poul-Henning Kamp. 2014. Please Put OpenSSL Out of Its Misery: OpenSSL must die, for it will never get any better. Queue 12, 3 (March 2014), 20–23. *https://doi.org/10.1145/2602649.2602816*

Seriously, you though Git was bad...

Lets download Bristol's certificate!

```
openssl s_client -connect bristol.ac.uk:443 -servername bristol.ac.uk
```

```
CONNECTED(00000003)
---
Certificate chain
 0 s:C=GB, ST=Bristol, City of, O=University of Bristol, CN=www.bristol.ac.uk
   i:C=NL, O=GEANT Vereniging, CN=GEANT OV RSA CA 4
   a:PKEY: rsaEncryption, 2048 (bit); sigalg: RSA-SHA384
   v:NotBefore: Sep 20 00:00:00 2024 GMT; NotAfter: Oct 13 23:59:59 2025 GMT
 1 s:C=NL, O=GEANT Vereniging, CN=GEANT OV RSA CA 4
   i:C=US, ST=New Jersey, L=Jersey City, O=The USERTRUST Network, CN=USERTrust RSA Certification Au
   a:PKEY: rsaEncryption, 4096 (bit); sigalg: RSA-SHA384
   v:NotBefore: Feb 18 00:00:00 2020 GMT; NotAfter: May 1 23:59:59 2033 GMT
 2 s:C=US, ST=New Jersey, L=Jersey City, O=The USERTRUST Network, CN=USERTrust RSA Certification Au
   i:C=GB, ST=Greater Manchester, L=Salford, O=Comodo CA Limited, CN=AAA Certificate Services
   a:PKEY: rsaEncryption, 4096 (bit); sigalg: RSA-SHA384
   v:NotBefore: Mar 12 00:00:00 2019 GMT; NotAfter: Dec 31 23:59:59 2028 GMT
---
Server certificate
-----BEGIN CERTIFICATE-----
MIIHoDCCBYigAwIBAgIQUdlcS7a9DzUEvZqekP9e1DANBgkqhkiG9w0BAQwFADBE
MQswCQYDVQQGEwJOTDEZMBcGA1UEChMQR0VBTlQgVmVyZW5pZ2luZzEaMBgGA1UE
AxMRR0VBTlQgT1YgUlNBIENBIDQwHhcNMjQwOTIwMDAwMDAwWhcNMjUxMDEzMjM1
OTU5WjBkMQswCQYDVQQGEwJHQjEZMBcGA1UECBMQQnJpc3RvbCwgQ2l0eSBvZjEe
MBwGA1UEChMVVW5pdmVyc2l0eSBvZiBCcmlzdG9sMRowGAYDVQQDExF3d3cuYnJp
c3RvbC5hYy51azCCASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQoCggEBAN0P6CdH
pbZyTsdd7mC5trQLRixBVA8f5Hbkjkx5SdUSHaMHxo7wdgq6+Mdpg0VZRoYYZZ0v
d7fHaw0UCho0X8Kc1TWOqzABzhO8KMrmp7+pYaCWaeqvai+c0JFug4OAGfOGNIVr
```

Before finally...

```
tlw9zLhol3ivph53CoTp8vi/OUcq0s8tU/G3mkNxICTRkmno
-----END CERTIFICATE-----
subject=C=GB, ST=Bristol, City of, O=University of Bristol, CN=www.bristol.ac.uk
issuer=C=NL, O=GEANT Vereniging, CN=GEANT OV RSA CA 4
---
No client certificate CA names sent
Peer signing digest: SHA256
Peer signature type: RSA
Server Temp Key: ECDH, prime256v1, 256 bits
---
SSL handshake has read 5657 bytes and written 450 bytes
Verification: OK
---
New, TLSv1.2, Cipher is ECDHE-RSA-AES128-GCM-SHA256
Protocol: TLSv1.2
Server public key is 2048 bit
Secure Renegotiation IS supported
Compression: NONE
Expansion: NONE
No ALPN negotiated
SSL-Session:
   Protocol : TLSv1.2
   Cipher : ECDHE-RSA-AES128-GCM-SHA256
   Session-ID: 3A507CAB0E14710F23192CE1765D1894A39BDF2E0A45CE8BB74D73A446ED68A7
   Session-ID-ctx:
   Master-Key: 5C3BC35E8611C315D4B763229604CAA10272685970693ACA6F7B478AFD46D88A422A8C6E4B35419B40AC
   PSK identity: None
   PSK identity hint: None
```

No but seriously this time

```
    SRP username: None
    Start Time: 1736955465
    Timeout : 7200 (sec)
    Verify return code: 0 (ok)
    Extended master secret: yes
---
```

Phew! The interesting bit was way back at the start though.

- Incidentally if you use the `openssl s_client` like this it'll let you communicate with TLS'd servers like you would with `netcat` or `telnet`
    - This will be useful to precisely 3 of you
        - At some point much later in your careers
        - When trying to debug why a network protocol isn't working
        - At 3am

Certificate chains

```
---
Certificate chain
 0 s:C=GB, ST=Bristol, City of, O=University of Bristol, CN=www.bristol.ac.uk
   i:C=NL, O=GEANT Vereniging, CN=GEANT OV RSA CA 4
   a:PKEY: rsaEncryption, 2048 (bit); sigalg: RSA-SHA384
   v:NotBefore: Sep 20 00:00:00 2024 GMT; NotAfter: Oct 13 23:59:59 2025 GMT
 1 s:C=NL, O=GEANT Vereniging, CN=GEANT OV RSA CA 4
   i:C=US, ST=New Jersey, L=Jersey City, O=The USERTRUST Network, CN=USERTrust RSA Certification Au
   a:PKEY: rsaEncryption, 4096 (bit); sigalg: RSA-SHA384
   v:NotBefore: Feb 18 00:00:00 2020 GMT; NotAfter: May 1 23:59:59 2033 GMT
 2 s:C=US, ST=New Jersey, L=Jersey City, O=The USERTRUST Network, CN=USERTrust RSA Certification Au
   i:C=GB, ST=Greater Manchester, L=Salford, O=Comodo CA Limited, CN=AAA Certificate Services
   a:PKEY: rsaEncryption, 4096 (bit); sigalg: RSA-SHA384
   v:NotBefore: Mar 12 00:00:00 2019 GMT; NotAfter: Dec 31 23:59:59 2028 GMT
---
```

0!

```
0 s:C=GB, ST=Bristol, City of, O=University of Bristol, CN=www.bristol.ac.uk
 i:C=NL, O=GEANT Vereniging, CN=GEANT OV RSA CA 4
 a:PKEY: rsaEncryption, 2048 (bit); sigalg: RSA-SHA384
 v:NotBefore: Sep 20 00:00:00 2024 GMT; NotAfter: Oct 13 23:59:59 2025 GMT
```

Our key is the 0th in the chain

- It's subject (s) is the organization University of Bristol, with the common name `www.bristol.ac.uk`
- It's issuer (i) is GEANT Vereiging (who are a European Reasearch and Education network)
- The algorithm (a) is 2048bit RSA with SHA384 signatures
- It is valid (v) from 2024-09-20 – 2025-10-13 (~one year)

Well I've never heard of them...

How many people have ever heard of GEANT before today?

- How do we know that they've checked that this certificate really belongs to the University of Bristol?

```
1 s:C=NL, O=GEANT Vereniging, CN=GEANT OV RSA CA 4
   i:C=US, ST=New Jersey, L=Jersey City, O=The USERTRUST Network, CN=USERTrust RSA Certification Au
   a:PKEY: rsaEncryption, 4096 (bit); sigalg: RSA-SHA384
   v:NotBefore: Feb 18 00:00:00 2020 GMT; NotAfter: May 1 23:59:59 2033 GMT
```

The next certificate in the chain validates who they are

- GEANT have a certificate that says that their identity has been validated by The USERTRUST Network
- Their certificate is valid for 13 years

Phew!

Well I've never heard of them…

How many people have ever heard of the USERTRUST Network before today?

- They can't even decide how to capitalize their name!
- How do we know that they've checked that its really GEANT who've issued our certificate?

```
2 s:C=US, ST=New Jersey, L=Jersey City, O=The USERTRUST Network, CN=USERTrust RSA Certification Aut
  i:C=GB, ST=Greater Manchester, L=Salford, O=Comodo CA Limited, CN=AAA Certificate Services
  a:PKEY: rsaEncryption, 4096 (bit); sigalg: RSA-SHA384
  v:NotBefore: Mar 12 00:00:00 2019 GMT; NotAfter: Dec 31 23:59:59 2028 GMT
```

Well they've been checked out by Comodo CA Limited…

- Well I've never heard of them…
- Whose certificate is valid for ~10 years

Well who am I to know who to trust...

On everyones computer is a root of trust: a set of certificates from people our system believes are able to authenticate others identities

- On UNIX-y systems its in `/etc/ssl/certs/ca-certificates.crt`
- Lists issued by OpenSSL, Mozilla, and others...

How many authorities are in it?

```
openssl storeutl -noout -text -certs /etc/ssl/certs/ca-certificates.crt | grep -c Subject:
```

```
149
```

```
openssl storeutl -noout -text -certs /etc/ssl/certs/ca-certificates.crt | grep Subject: | grep 'Co
```

- Subject: *C=GB ST=Greater Manchester L=Salford O=Comodo CA Limited CN=AAA Certificate Services*

Chains of trust

So since:

My system trusts Comodo CA Limited
Commodo has checked USERTRUST
USERTRUST has checked Geant
Geant has checked the University of Bristol

This must be good right?

- Anyone want to suggest a problem?

Problems

- How exactly did they check them?
- 149 is a lot of trusted entities...
- How do you get on this list in the first place?
- 10 years is a long time to keep something secure for...
- Don't you have to pay for these certificates?

How did they check them?

Well... it depends on what you paid

- Most Certificate Authorities have multiple levels of identity validation
  - For a little they'll just sign you a certificate
  - For a lot they'll just sign you a certificate and check your identity

Have Certificate Authorities lost their keys?

Yes.

Have Certificate Authorities signed things they shouldn't have?

Yes.

Have Certificate Authorities signed things when governments ask them to?

Yes.

Is it all a bit of a racket?

Yes.

So what did people do?

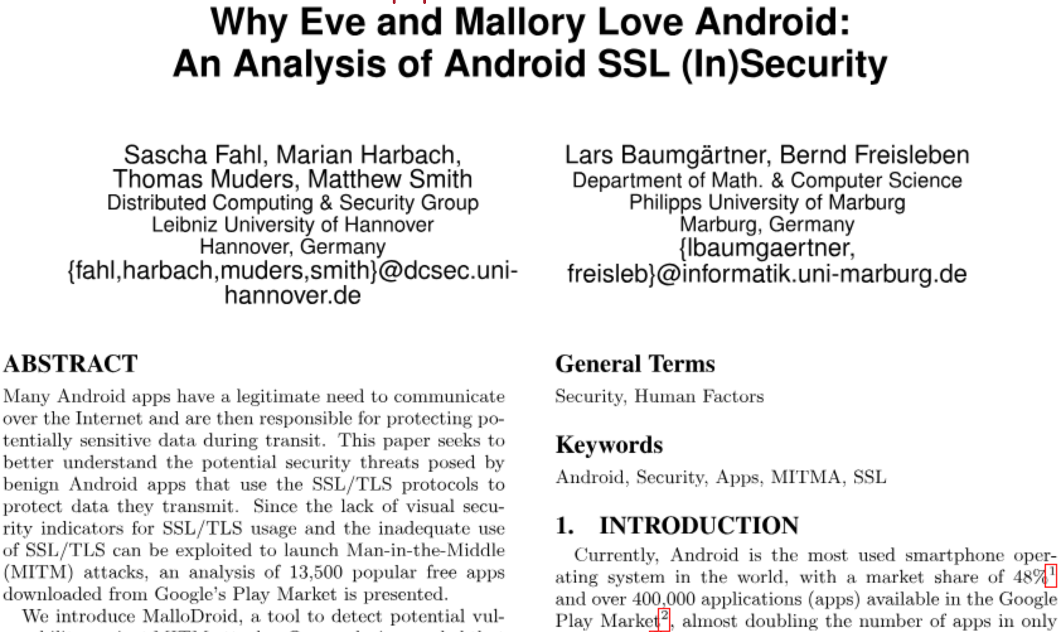
If you're running a personal website you could just sign your own certificate

- The browser would say that "it isn't signed by a trusted authority"...
- Still better than nothing (at least the traffic is still encrypted, even if you can't validate it belongs to the right person).

Except...

We don't just use certificates for webpages...

In 2012 Sascha Fahl writes this paper…



Why Eve and Mallory Love Android: An Analysis of Android SSL (In)Security

Sascha Fahl, Marian Harbach,
Thomas Muders, Matthew Smith
Distributed Computing & Security Group
Leibniz University of Hannover
Hannover, Germany
{fahl,harbach,muders,smith}@dcsec.uni-hannover.de

Lars Baumgärtner, Bernd Freisleben
Department of Math. & Computer Science
Philipps University of Marburg
Marburg, Germany
{lbaumgaertner, freisleb}@informatik.uni-marburg.de

ABSTRACT

Many Android apps have a legitimate need to communicate over the Internet and are then responsible for protecting potentially sensitive data during transit. This paper seeks to better understand the potential security threats posed by benign Android apps that use the SSL/TLS protocols to protect data they transmit. Since the lack of visual security indicators for SSL/TLS usage and the inadequate use of SSL/TLS can be exploited to launch Man-in-the-Middle (MITM) attacks, an analysis of 13,500 popular free apps downloaded from Google's Play Market is presented.

We introduce MalloDroid, a tool to detect potential vul-

General Terms

Security, Human Factors

Keywords

Android, Security, Apps, MITMA, SSL

1. INTRODUCTION

Currently, Android is the most used smartphone operating system in the world, with a market share of 48%[1] and over 400,000 applications (apps) available in the Google Play Market[2], almost doubling the number of apps in only

So what's it all about...

In 2012 apps on your phone are cool!

- Everyone is writing them
- Developers are selling them for free
- Subscriptions weren't a thing yet
- Apps are providing native access to web services

To avoid having errors in their apps...

- Thousands have turned off certificate warnings
- Thousands have misconfigured TLS
- Security people are moaning that these apps are sending personal data
  - But GDPR/EU regulations aren't a thing yet...

Why is this bad?

Suppose I'm sitting on your network connection

- If I can intercept your DNS traffic (what turns websites into phone numbers) I can redirect your web traffic
- If I can redirect your web traffic I can look at what is in it and modify it (unless its encrypted)
- If I created a fake certificate I can decrypt you traffic, and forward it to the real website and hijakc what you do
  - But you'll see a warning saying Bad certificate
    - ...unless you've turned off warnings.

This is a man-in-the-middle attack!

What is the response?

1. Lets disable the option to disable certificate checking!
2. Lets introduce certificate pinning to prevent websites changing their certificates before the expiry date!
3. Lets make getting an SSL certificate free

Lets Encrypt!

The worlds biggest certificate authority
Founded by Mozilla, The EFF and the Linux Foundation (amongst others), to make encryption ubiquitous...
They will issue anyone a certificate for free provided you can complete their ACME protocol

ACME

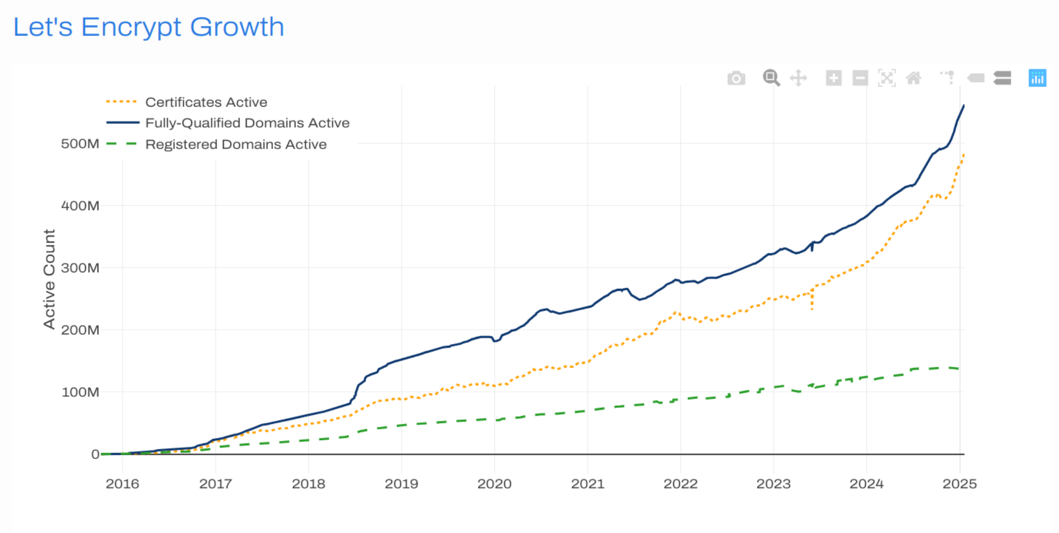
ACME Protocol (Version 2.0) standardized in RFC 8555

1. You enrol your public key with an ACME server and say that it will be used to manage your domain
2. You create your own certificate and ask the ACME server to sign it
3. ACME server gives you a challenge and asks you to sign it with the key you enrolled
4. If you can host that signature at a `.well-known/acme-challenge/` address on your server, they'll give you a short-lived TLS certificate
5. Or if you can put that signature in your DNS record they'll give you a certificate

ACME clients automate the certificate request and update process

- Full security model in RFC 8555
- Provided your key stays secure, then the same person is at least running the same server
- If it is compromised then at least they certificate won't be valid for a long time

It's been wildly successful...



Let's Encrypt Growth

Public Key Cryptography and TLS

Massive success story

- Most web traffic is now encrypted
- If you're making a new web service… encrypt it!

Still security concerns…

- But mostly it just works?

Public Key Cryptography FTW!!

So far!

- We've talked about what public key cryptography is...
- We've talked about how it can be used as an access control system with OpenSSH
- We've talked about how it can be used to authenticate and encrypt traffic to webservers with OpenSSL

Lets move on to authenticating people!

PGP

Pretty Good Privacy developed by Phil Zimmerman in 1996

- Generic framework for doing stuff with public key cryptography
- Typically used for verifying downloads and email encryption
  - But can do loads of things...

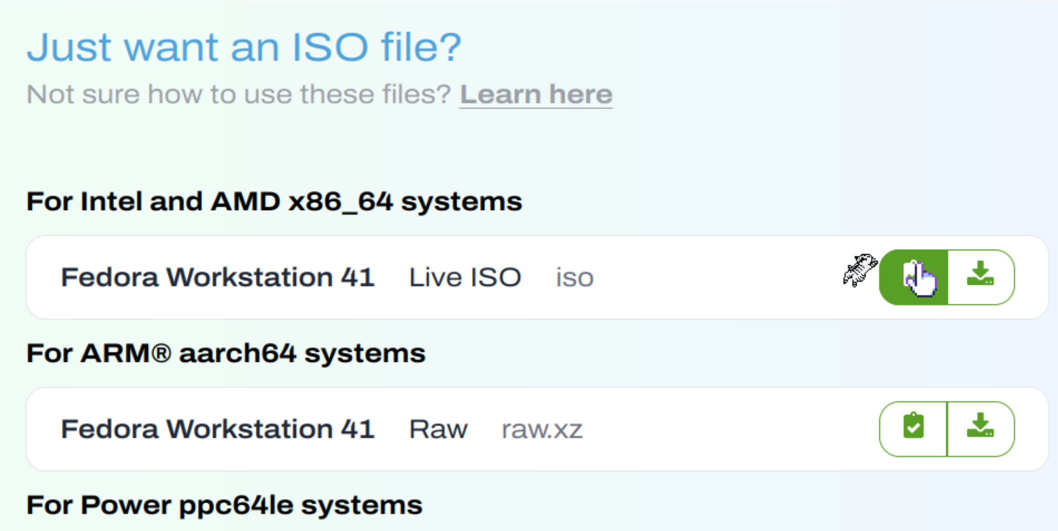
It is illegal in some countries!

- If your country has laws about strong encryption it usually means PGP
- Some countries don't like crypto software being exported

For example

Say you want to download a new OS...

- If you go to Fedora (like Debian but slightly different culture) you'll find a download and a verify button



Just want an ISO file?

Not sure how to use these files? Learn here

For Intel and AMD x86_64 systems

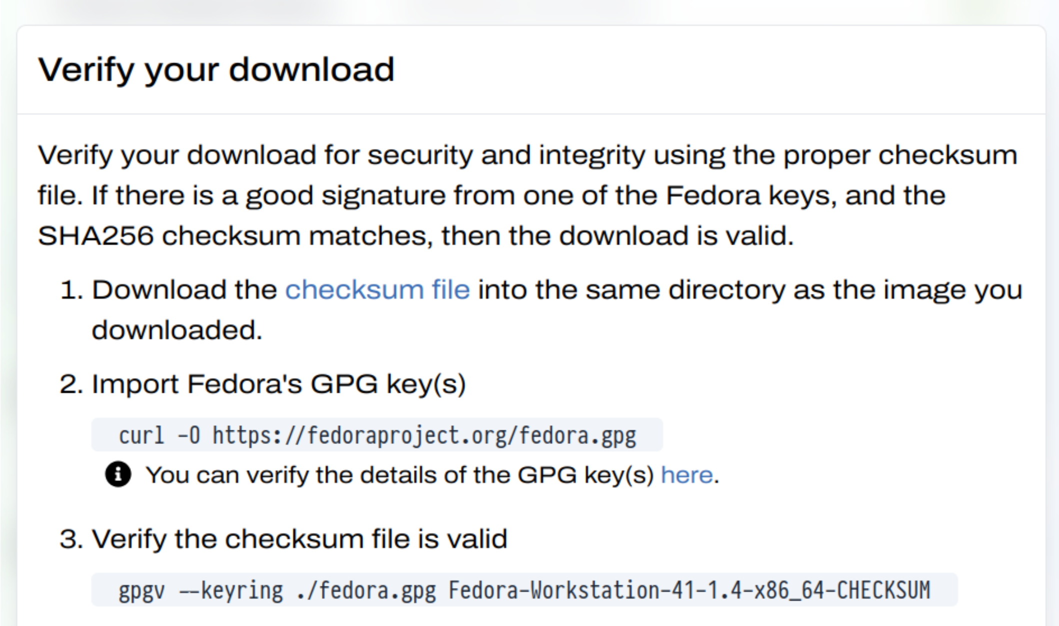
Fedora Workstation 41 Live ISO iso

For ARM® aarch64 systems

Fedora Workstation 41 Raw raw.xz

For Power ppc64le systems

And verify



Verify your download

Verify your download for security and integrity using the proper checksum file. If there is a good signature from one of the Fedora keys, and the SHA256 checksum matches, then the download is valid.

1. Download the checksum file into the same directory as the image you downloaded.
2. Import Fedora's GPG key(s)

```
curl -O https://fedoraproject.org/fedora.gpg
```

ℹ You can verify the details of the GPG key(s) here.

3. Verify the checksum file is valid

```
gpgv --keyring ./fedora.gpg Fedora-Workstation-41-1.4-x86_64-CHECKSUM
```

What about Git?!

How do you know who really made a Git commit?

```
git log --pretty -1
```

```
commit beec8cf4b4835bcb42f76bed75c96af15d07249a
Author: Joseph Hallett <joseph.hallett@bristol.ac.uk>
Date:   Tue Jan 21 17:10:01 2025 +0000

    Getting 2025's slides started
```

I can't even remember what day it is...

Sure it says it's me...

- But its just whatever you stick in `git config user.name`...
- I could just as easily make it say someone else

Lets sign our commits with our public key!

```
git commit -S -a -m "Signing␣a␣commit␣for␣demo␣purposes!"
```

```
[main b38373c] Signing a commit for demo purposes!
    3 files changed, 17 insertions(+), 2 deletions(-)
```

And when you view the log...'

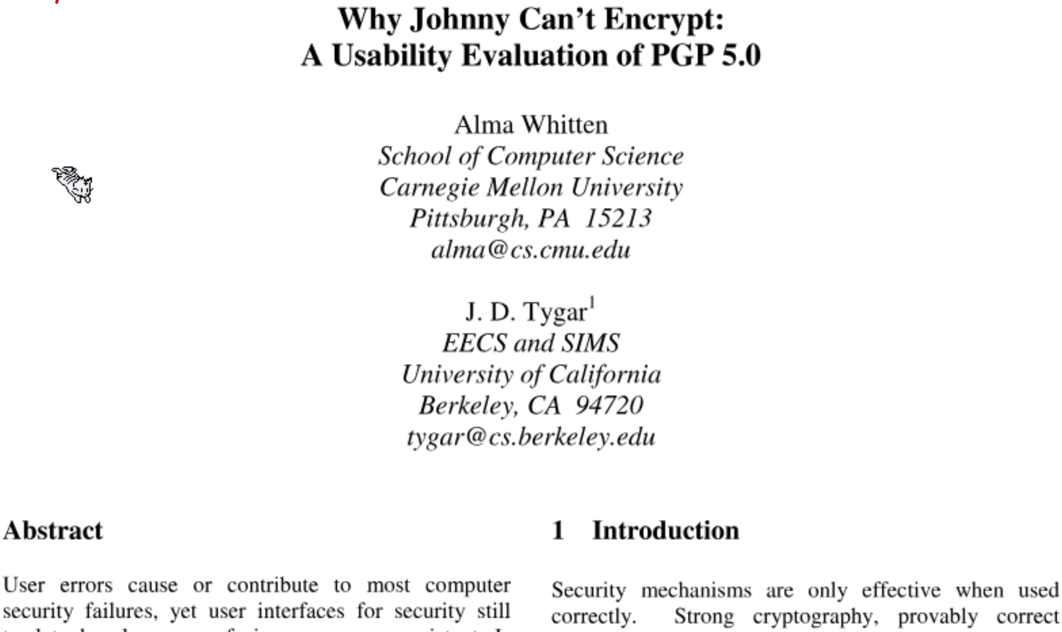
```
git log --pretty --show-signature -1
```

```
commit b38373c1c56c1d20b5152424a2a7c8fc5b46a070
gpg: Signature made Wed 22 Jan 2025 12:03:57 GMT
gpg:                using RSA key 364B7926504D5645137778801413109AF27CBFAF9
gpg: Good signature from "Joseph Hallett <joseph.hallett@bristol.ac.uk>" [ult
Primary key fingerprint: 364B 7926 504D 5645 1377  7801 4131 09AF 27CB FAF9
Author: Joseph Hallett <joseph.hallett@bristol.ac.uk>
Date:   Wed Jan 22 12:03:57 2025 +0000

    Signing a commit for demo purposes!
```

If you use GitHub and set you key up it'll even display a little verified button...

It's easy to use!



Why Johnny Can't Encrypt: A Usability Evaluation of PGP 5.0

Alma Whitten
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213
alma@cs.cmu.edu

J. D. Tygar[1]
EECS and SIMS
University of California
Berkeley, CA 94720
tygar@cs.berkeley.edu

Abstract

User errors cause or contribute to most computer security failures, yet user interfaces for security still

1 Introduction

Security mechanisms are only effective when used correctly. Strong cryptography, provably correct

The classic study in security usability

Security software is really hard to use!

- We should do better
- It's not the users fault: the software is bad

If you see a paper called Why Johnny can't… its referencing this…

I hate this study…

I mean it is not wrong

- But its overstated

UNIX program running as a Mac App with Windows users

- …and we wonder why people couldn't use a complex tool

I don't think its that bad

- …but I like Git…
- It's definitely confusing
- The docs aren't great

Practice makes perfect

We're going to explore GPG in the lab this week!

- We're going to get you set up with keys and get you sending email and verifying signatures…

But there are some concepts I want to go over first

Set up...

Like all other public key systems you have public and private keys

- But how do you trust that a key belongs to someone?

With OpenSSH

- We fell back to passwords

With OpenSSL

- We used a root of trust

With PGP

- We use a web of trust

Web of Trust

With OpenSSL we have a centralized source of trust in the `ca-certificates.pem` bundle…
With PGP we have decentralized trust

- You trust yourself (hopefully)
- If someone checks you they sign your key and give you a signature
- If you check someone else you sign their key and give them a signature

When checking other people's keys you

- Check if you trust them
- Check if you trust someone who trusts them

Party time

To build this web of trust you need to build a community of other PGP users

- But how do you do that?

Key signing party!

- Used to be common at security conferences
- Everyone gets together and signs everyone else's keys!

How do you even get other peoples keys?

Various organizations run keyservers where you can store your keys...

- e.g. `https://keyserver.ubuntu.com`

You can fetch other people's keys from here...

- But maybe don't trust them until you can verify them?

Conclusions

We've covered:

- Public key crypto fundamentals
- Explained how SSH auth works
- Talked about OpenSSL and certificate authorities
- Talked about PGP and the web of trust

In the lab:

- PGP keys!