Security Engineering (5)

Email: christian.urban at kcl.ac.uk
Office: N7.07 (North Wing, Bush House)
Slides: KEATS (also homework is there)

Topical Slide

- Protocoll attack against Wifi clients
- you can force a client to install choosen keys (000...000)
- all Unix-based devices are affected (Windows not so much, since they do not fully implement the Wifi standard)

Protocols

- Other examples: Wifi, Http-request, TCP-request, card readers, RFID (passports)...

Protocols

- Other examples: Wifi, Http-request, TCP-request, card readers, RFID (passports)...
- The point is that we cannot control the network: An attacker can install a packet sniffer, inject packets, modify packets, replay messages...fake pretty much everything.

Keyless Car Transponders

- There are two security mechanisms: one remote central locking system and one passive RFID tag (engine immobiliser).
- How can I get in? How can thieves be kept out? How to avoid MITM attacks?

Papers: Gone in 360 Seconds: Hijacking with Hitag2,
Dismantling Megamos Crypto: Wirelessly Lockpicking
a Vehicle Immobilizer

Problems with Key Fobs

MONDAY 27 OCTOBER 2014 EVENING STANDARD

:ebook.com/eveningstandard
us on Twitter @standardnews

Insurers refuse to cover Range Rovers due to security flaw

Kiran Randhawa

INSURANCE companies are refusing to cover new Range Rovers in London after thieves found a way of bypassing the vehicles' keyless ignition systems.

Criminals use hand-held electronic devices, available on eBay, to get around the feature. Unless owners have secure parking, underwriters are now said to be refusing to insure them.

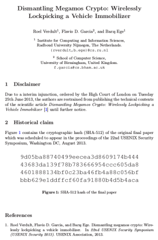
Insurers have asked to meet Jaguar Range Rover to discuss the growing problem. Thatcham Research, the motor insurers' automotive research centre, said that 294 Range Rover Evoque and Sport vehicles were stolen in London between January and July. In the same period, 63 BMW X5s, a rival to the Range Rover, were taken.

James Wasdell, co-founder of Quantum Underwriting, said: "If you are an owner of a street-parked Range Rover, nine out of 10 insurers will now say no. However, we have found a solution by combining the use of physical protection [for the car] and advising clients to insure all assets with one insurer."

Jaguar Land Rover said: "Our line-up continues to meet the insurance industry requirements. Nevertheless we are taking this issue very seriously."

Circumventing the ignition protection:

- either dismantling Megamos crypto,
- or use the diagnostic port to program blank keys



Dismantling Megamos Crypto: Wirelessly Lockpicking a Vehicle Immobilizer

Roel Verdult[1], Flavio D. Garcia[2], and Barış Ege[1]

[1] Institute for Computing and Information Sciences, Radboud University Nijmegen, The Netherlands. {rverdult,b.ege}@cs.ru.nl

[2] School of Computer Science, University of Birmingham, United Kingdom. f.garcia@cs.bham.ac.uk

1 Disclaimer

Due to a interim injunction, ordered by the High Court of London on Tuesday 25th June 2013, the authors are restrained from publishing the technical contents of the scientific article *Dismantling Megamos Crypto: Wirelessly Lockpicking a Vehicle Immobilizer* [1] until further notice.

2 Historical claim

Figure 1 contains the cryptographic hash (SHA-512) of the original final paper which was scheduled to appear in the proceedings of the 22nd USENIX Security Symposium, Washington DC, August 2013.

```
9d05ba88740499eecea3d8609174b444
43683da139f78b783666954ccc605da8
4601888134bf0c23ba46fb4a88c056bf
bbb629e1ddffcf60fa91880b4d5b4aca
```

Figure 1: SHA-512 hash of the final paper

References

1. Roel Verdult, Flavio D. Garcia, and Barış Ege. Dismantling megamos crypto: Wirelessly lockpicking a vehicle immobilizer. In *22nd USENIX Security Symposium (USENIX Security 2013)*. USENIX Association, 2013.

HTTPS / GSM

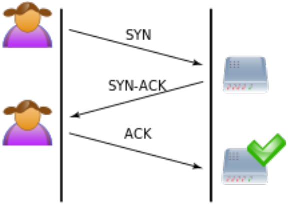
- I am sitting at Starbuck. How can I be sure I am really visiting Barclays? I have no control of the access point.
- How can I achieve that a secret key is established in order to encrypt my mobile conversation? I have no control over the access points.

G2o Summit in 2009

- Snowden documents reveal "that during the G20 meetings...GCHQ used 'ground-breaking intelligence capabilities' to intercept the communications of visiting delegations. This included setting up internet cafes where they used an email interception program and key-logging software to spy on delegates' use of computers..."
- "The G20 spying appears to have been organised for the more mundane purpose of securing an advantage in meetings."

Handshakes

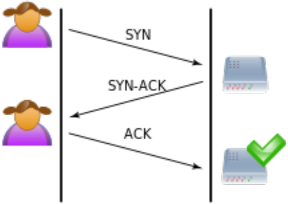
- starting a TCP connection between a client and a server initiates the following three-way handshake protocol:



Alice:	Hello server!
Server:	I heard you
Alice:	Thanks

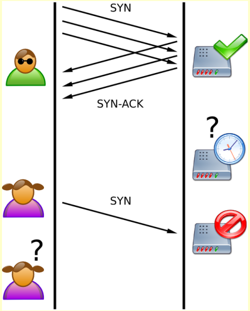
Handshakes

- starting a TCP connection between a client and a server initiates the following three-w handshake protocol:



Alice:
Server:
Alice:

SYNflood attacks:



Protocols

$$A \rightarrow B : \ldots$$

- by convention A, B are named principals Alice... but most likely they are programs, which just follow some instructions (they are more like roles)

Protocols

$$\begin{array}{rcl} A & \rightarrow & B : \ldots \\ B & \rightarrow & A : \ldots \\ & \vdots & \end{array}$$

- by convention A, B are named principals Alice... but most likely they are programs, which just follow some instructions (they are more like roles)

- indicates one "protocol run", or session, which specifies some order in the communication
- there can be several sessions in parallel (think of wifi routers)

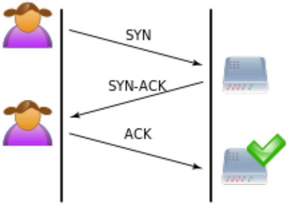
Messages

$$A \rightarrow B : msg$$

- Unencrypted: msg
- Random number (nonce): N
- Encrypted: $\{msg\}_K$, $\{msg_1, msg_2\}_K$, $\{\{msg\}_{K_1}\}_{K_2}$

Handshakes

- starting a TCP connection between a client and a server initiates the following three-way handshake protocol:



Alice:	Hello server!
Server:	I heard you
Alice:	Thanks

$$A \to S: \quad \text{SYN}$$
$$S \to A: \quad \text{SYN-ACK}$$
$$A \to S: \quad \text{ACK}$$

Cryptographic Protocol Failures

Ross Anderson and Roger Needham wrote:

> A lot of the recorded frauds were the result of this kind of blunder, or from management negligence pure and simple. However, there have been a significant number of cases where the designers protected the right things, used cryptographic algorithms which were not broken, and yet found that their systems were still successfully attacked.

Oyster Cards

- good example of a bad protocol (security by obscurity)

Wirelessly Pickpocketing a Mifare Classic Card

The Mifare Classic is the most widely used contactless smartcard on the market. The stream cipher CRYPTO1 used by the Classic has recently been reverse engineered and serious attacks have been proposed. The most serious of them retrieves a secret key in under a second. In order to clone a card, previously proposed attacks require that the adversary either has access to an eavesdropped communication session or executes a message-by-message man-in-the-middle attack between the victim and a legitimate reader. Although this is already disastrous from a cryptographic point of view, system integrators maintain that these attacks cannot be performed undetected.

This paper proposes four attacks that can be executed by an adversary having only wireless access to just a card (and not to a legitimate reader). The most serious of them recovers a secret key in less than a second on ordinary hardware. Besides the cryptographic weaknesses, we exploit other weaknesses in the protocol stack. A vulnerability in the computation of parity bits allows an adversary to establish a side channel. Another vulnerability regarding nested authentications provides enough plaintext for a speedy known-plaintext attack. (a paper from 2009)

Oyster Cards

- good example of a bad protocol (security by obscurity)
- *"Breaching security on Oyster cards should not allow unauthorised use for more than a day, as TfL promises to turn off any cloned cards within 24 hours..."*

Another Example

In an email from Ross Anderson

From: Ross Anderson <Ross.Anderson@cl.cam.ac.uk>
Sender: cl-security-research-bounces@lists.cam.ac.uk
To: cl-security-research@lists.cam.ac.uk
Subject: Birmingham case
Date: Tue, 13 Aug 2013 15:13:17 +0100

As you may know, Volkswagen got an injunction against the University of Birmingham suppressing the publication of the design of a weak cipher used in the remote key entry systems in its recent-model cars. The paper is being given today at Usenix, minus the cipher design.

I've been contacted by Birmingham University's lawyers who seek to prove that the cipher can be easily obtained anyway. They are looking for a student who will download the firmware from any newish VW, disassemble it and look for the cipher. They'd prefer this to be done by a student rather than by a professor to emphasise how easy it is.

Volkswagen's argument was that the Birmingham people had reversed a locksmithing tool produced by a company in Vietnam, and since their key fob chip is claimed to be tamper-resistant, this must have involved a corrupt insider at VW or at its supplier Thales. Birmingham's argument is that this is nonsense as the cipher is easy to get hold of. Their lawyers feel this argument would come better from an independent outsider.

Let me know if you're interested in having a go, and I'll put you in touch
Ross

Authentication Protocols

Alice (A) and Bob (B) share a secret key K_{AB}

Passwords:

$$B \rightarrow A : K_{AB}$$

Authentication Protocols

Alice (A) and Bob (B) share a secret key K_{AB}

Passwords:

$$B \to A : K_{AB}$$

Problem: Eavesdropper can capture the secret and replay it; A cannot confirm the identity of B

Authentication?

"On the Internet, nobody knows you're a dog."

Authentication Protocols

Alice (A) and Bob (B) share a secret key K_{AB}

Simple Challenge Response:

$$\begin{array}{ll} A \rightarrow B : & N \\ B \rightarrow A : & \{N\}_{K_{AB}} \end{array}$$

Authentication Protocols

Alice (A) and Bob (B) share a secret key K_{AB}

Mutual Challenge Response:

$$
\begin{aligned}
A \rightarrow B &: \quad N_A \\
B \rightarrow A &: \quad \{N_A, N_B\}_{K_{AB}} \\
A \rightarrow B &: \quad N_B
\end{aligned}
$$

Nonces

1. I generate a nonce (random number) and send it to you encrypted with a key we share
2. you increase it by one, encrypt it under a key I know and send it back to me

I can infer:

- you must have received my message
- you could only have generated your answer after I send you my initial message
- if only you and me know the key, the message must have come from you

$$A \to B: \quad N_A$$
$$B \to A: \quad \{N_A, N_B\}_{K_{AB}}$$
$$A \to B: \quad N_B$$

The attack (let A decrypt her own messages):

$$A \to E: \quad N_A$$
$$E \to A: \quad N_A$$
$$A \to E: \quad \{N_A, N'_A\}_{K_{AB}}$$
$$E \to A: \quad \{N_A, N'_A\}_{K_{AB}}$$
$$A \to E: \quad N'_A \ \ (= N_B)$$

$$
\begin{aligned}
A \rightarrow B &: \quad N_A \\
B \rightarrow A &: \quad \{N_A, N_B\}_{K_{AB}} \\
A \rightarrow B &: \quad N_B
\end{aligned}
$$

The attack (let A decrypt her own messages):

$$
\begin{aligned}
A \rightarrow E &: \quad N_A \\
E \rightarrow A &: \quad N_A \\
A \rightarrow E &: \quad \{N_A, N'_A\}_{K_{AB}} \\
E \rightarrow A &: \quad \{N_A, N'_A\}_{K_{AB}} \\
A \rightarrow E &: \quad N'_A \quad (= N_B)
\end{aligned}
$$

Solutions: $K_{AB} \neq K_{BA}$ or include an id in the second message

Encryption to the Rescue?

- $A \to B : \{A, N_A\}_{K_{AB}}$ encrypted
- $B \to A : \{N_A, K'_{AB}\}_{K_{AB}}$
- $A \to B : \{N_A\}_{K'_{AB}}$

Encryption to the Rescue?

- $A \rightarrow B : \{A, N_A\}_{K_{AB}}$ encrypted
- $B \rightarrow A : \{N_A, K'_{AB}\}_{K_{AB}}$
- $A \rightarrow B : \{N_A\}_{K'_{AB}}$

  means you need to send separate “Hello” signals (bad), or worse share a single key between many entities

Public-Key Infrastructure

- the idea is to have a certificate authority (CA)
- you go to the CA to identify yourself
- CA: "I, the CA, have verified that public key P^{pub}_{Bob} belongs to Bob"

- CA must be trusted by everybody
- What happens if CA issues a false certificate? Who pays in case of loss? (VeriSign explicitly limits liability to $100.)

A Simple PK Protocol

"Normal" protocol run:

- *A* sends public key to *B*
- *B* sends public key to *A*
- *A* sends message encrypted with *B*'s public key, *B* decrypts it with its private key
- *B* sends message encrypted with *A*'s public key, *A* decrypts it with its private key

A Simple PK Protocol

1. $A \rightarrow B : K_A^{pub}$
2. $B \rightarrow A : K_B^{pub}$
3. $A \rightarrow B : \{A, m\}_{K_B^{pub}}$
4. $B \rightarrow A : \{B, m'\}_{K_A^{pub}}$

A Simple PK Protocol

1. $A \to B : K_A^{pub}$
2. $B \to A : K_B^{pub}$
3. $A \to B : \{A, m\}_{K_B^{pub}}$
4. $B \to A : \{B, m'\}_{K_A^{pub}}$

unfortunately there is a simple man-in-the-middle-attack

Man-in-the-Middle

Attack:

- A sends public key to B — C intercepts this message and send his own public key
- B sends public key to A — C intercepts this message and send his own public key
- A sends message encrypted with C's public key, C decrypts it with its private key, re-encrypts with B's public key
- similar for other direction

A MITM Attack

1. $A \rightarrow E : K_A^{pub}$
2. $E \rightarrow B : K_E^{pub}$
3. $B \rightarrow E : K_B^{pub}$
4. $E \rightarrow A : K_E^{pub}$
5. $A \rightarrow E : \{A, m\}_{K_E^{pub}}$
6. $E \rightarrow B : \{E, m\}_{K_B^{pub}}$
7. $B \rightarrow E : \{B, m'\}_{K_E^{pub}}$
8. $E \rightarrow A : \{E, m'\}_{K_A^{pub}}$

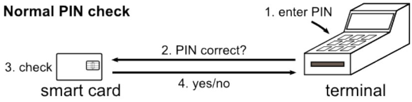
A MITM Attack

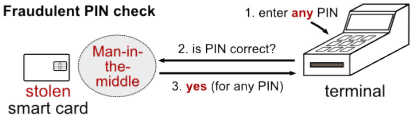
1. $A \rightarrow E : K_A^{pub}$
2. $E \rightarrow B : K_E^{pub}$
3. $B \rightarrow E : K_B^{pub}$
4. $E \rightarrow A : K_E^{pub}$
5. $A \rightarrow E : \{A, m\}_{K_E^{pub}}$
6. $E \rightarrow B : \{E, m\}_{K_B^{pub}}$
7. $B \rightarrow E : \{B, m'\}_{K_E^{pub}}$
8. $E \rightarrow A : \{E, m'\}_{K_A^{pub}}$

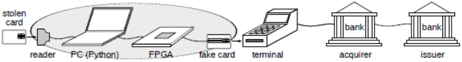
and A and B have no chance to detect it

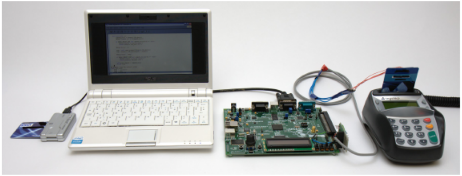
A Man-in-the-middle attack in real life:

- the card only says yes to the terminal if the PIN is correct
- trick the card in thinking transaction is verified by signature
- trick the terminal in thinking the transaction was verified by PIN









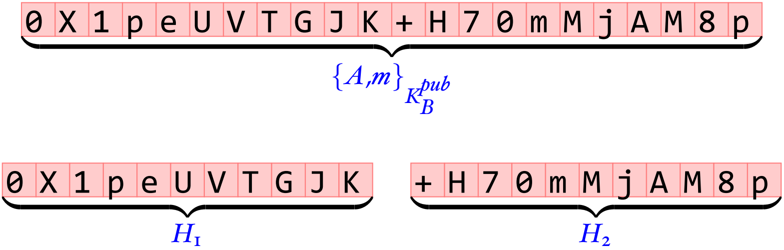
Interlock Protocol

The interlock protocol ("best bet" against MITM):

1. $A \to B : K_A^{pub}$
2. $B \to A : K_B^{pub}$
3. $\{A, m\}_{K_B^{pub}} \mapsto H_1, H_2$

   $\{B, m'\}_{K_A^{pub}} \mapsto M_1, M_2$
4. $A \to B : H_1$
5. $B \to A : \{H_1, M_1\}_{K_A^{pub}}$
6. $A \to B : \{H_2, M_1\}_{K_B^{pub}}$
7. $B \to A : M_2$

Splitting Messages



- you can also use the even and odd bytes
- the point is you cannot decrypt the halves, even if you have the key

$$A \to C : K_A^{pub}$$
$$C \to B : K_C^{pub}$$
$$B \to C : K_B^{pub}$$
$$C \to A : K_C^{pub}$$
$$\{A, m\}_{K_C^{pub}} \mapsto H_1, H_2$$
$$\{B, m'\}_{K_C^{pub}} \mapsto M_1, M_2$$
$$\{C, a\}_{K_B^{pub}} \mapsto C_1, C_2$$
$$\{C, b\}_{K_A^{pub}} \mapsto D_1, D_2$$

$$A \to C : H_1$$
$$C \to B : C_1$$
$$B \to C : \{C_1, M_1\}_{K_C^{pub}}$$
$$C \to A : \{H_1, D_1\}_{K_A^{pub}}$$
$$A \to C : \{H_2, D_1\}_{K_C^{pub}}$$
$$C \to B : \{C_2, M_1\}_{K_B^{pub}}$$
$$B \to C : M_2$$
$$C \to A : D_2$$

$$A \to C : K_A^{pub}$$
$$C \to B : K_C^{pub}$$
$$B \to C : K_B^{pub}$$
$$C \to A : K_C^{pub}$$

$$\{A, m\}_{K_C^{pub}} \mapsto H_1, H_2$$
$$\{B, m'\}_{K_C^{pub}} \mapsto M_1, M_2$$

$$\{C, a\}_{K_B^{pub}} \mapsto C_1, C_2$$
$$\{C, b\}_{K_A^{pub}} \mapsto D_1, D_2$$

$$A \to C : H_1$$
$$C \to B : C_1$$
$$B \to C : \{C_1, M_1\}_{K_C^{pub}}$$
$$C \to A : \{H_1, D_1\}_{K_A^{pub}}$$
$$A \to C : \{H_2, D_1\}_{K_C^{pub}}$$
$$C \to B : \{C_2, M_1\}_{K_B^{pub}}$$
$$B \to C : M_2$$
$$C \to A : D_2$$

m = How is your grandmother? m' = How is the weather today in London?

- you have to ask something that cannot be imitated (requires A and B know each other)
- what happens if m and m' are voice messages?

- you have to ask something that cannot be imitated (requires A and B know each other)
- what happens if m and m' are voice messages?

- So C can either leave the communication unchanged, or invent a complete new conversation

- the moral: establishing a secure connection from "zero" is almost impossible—you need to rely on some established trust
- that is why PKI relies on certificates, which however are badly, badly realised

Car Transponder (HiTag2)

1. C generates a random number N
2. C calculates $(F, G) = \{N\}_K$
3. $C \rightarrow T$: N, F
4. T calculates $(F', G') = \{N\}_K$
5. T checks that $F = F'$
6. $T \rightarrow C$: N, G'
7. C checks that $G = G'$

Car Transponder (HiTag2)

1. C generates a random number N
2. C calculates $(F, G) = \{N\}_K$
3. $C \to T$: N, F
4. T calculates $(F', G') = \{N\}_K$
5. T checks that $F = F'$
6. $T \to C$: N, G'
7. C checks that $G = G'$

This process means that the transponder believes the car knows the key K, and the car believes the transponder knows the key K. They have authenticated themselves to each other, or have they?

Trusted Third Parties

Simple protocol for establishing a secure connection via a mutually trusted 3rd party (server):

$$
\begin{aligned}
A &\rightarrow S : A, B \\
S &\rightarrow A : \{K_{AB}, \{K_{AB}\}_{K_{BS}}\}_{K_{AS}} \\
A &\rightarrow B : \{K_{AB}\}_{K_{BS}} \\
A &\rightarrow B : \{m\}_{K_{AB}}
\end{aligned}
$$

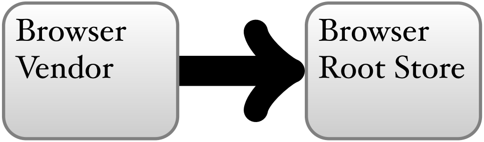
PKI: The Main Idea

- the idea is to have a certificate authority (CA)
- you go to the CA to identify yourself
- CA: "I, the CA, have verified that public key P^{pub}_{Bob} belongs to Bob"

- CA must be trusted by everybody

- certificates are time limited, and can be revoked
- What happens if CA issues a false certificate? Who pays in case of loss? (VeriSign explicitly limits liability to $100.)

PKI: Chains of Trust



- CAs make almost no money anymore, because of stiff competition
- browser companies are not really interested in security; only in market share

PKI: Weaknesses

CAs just cannot win (make any profit):

- there are hundreds of CAs, which issue millions of certificates and the error rate is small
- users (servers) do not want to pay or pay as little as possible

- a CA can issue a certificate for any domain not needing any permission (CAs are meant to undergo audits, but...DigiNotar)
- if a CA has issued many certificates, it "becomes too big to fail"
- Can we be sure CAs are not just frontends of some government organisation?

PKI: Weaknesses

- many certificates are issued via Whois, whether you own the domain...if you hijacked a domain, it is easy to obtain certificates
- the revocation mechanism does not work (Chrome has given up on general revocation lists)
- lax approach to validation of certificates (Have you ever bypassed certification warnings?)
- sometimes you want to actually install invalid certificates (self-signed)

PKI: Attacks

- Go directly after root certificates
  - governments can demand private keys
  - 10 years ago it was estimated that breaking a 1024 bit key takes one year and costs 10 - 30 Mio $; this is now reduced to 1 Mio $
- Go after buggy implementations of certificate validation
- Social Engineering
  - in 2001 somebody pretended to be from Microsoft and asked for two code-signing certificates

The eco-system is completely broken (it relies on thousands of entities to do the right thing). Maybe DNSSEC where keys can be attached to domain names is a way out.

Real Attacks

- In 2011, DigiNotar (Dutch company) was the first CA that got compromised comprehensively, and where many fraudulent certificates were issued to the wild. It included approximately 300,000 IP addresses, mostly located in Iran. The attackers (in Iran?) were likely interested "only" in collecting gmail passwords.

- The Flame malware piggy-bagged on this attack by advertising malicious Windows updates to some targeted systems (mostly in Iran, Israel, Sudan).

PKI is Broken

- PKI and certificates are meant to protect you against MITM attacks, but if the attack occurs your are presented with a warning and you need to decide whether you are under attack.
- Webcontent gets often loaded from 3rd-party servers, which might not be secured
- Misaligned incentives: browser vendors are not interested in breaking webpages with invalid certificates

Why are there so many invalid certificates?

- insufficient name coverage (www.example.com should include example.com)
- IoT: many appliances have web-based admin interfaces; the manufacturer cannot know under which IP and domain name the appliances are run (so cannot install a valid certificate)
- expired certificates, or incomplete chains of trust (servers are supposed to supply them)