OWASP
AppSec Europe
London 2nd-6th July 2018

Man in the contacts

and secure messengers: the ultimate spear-phishing weapon?

Laureline DAVID / Jérémy MATOS

Man in the contacts

Laureline DAVID / Jérémy MATOS

About the authors

Both based in French speaking Switzerland

Laureline DAVID

- Freelance consultant
- HEIG-VD graduate in Security Engineering
- https://ltouroumov.ch/

Jérémy MATOS

- Former developer of security solutions
- OWASP Geneva co-chapter leader
- Threat modeling aficionado
- Appsec consultant at my own company: Securing Apps
- https://www.securingapps.com/
- @SecuringApps

Man in the contacts

Laureline DAVID / Jérémy MATOS

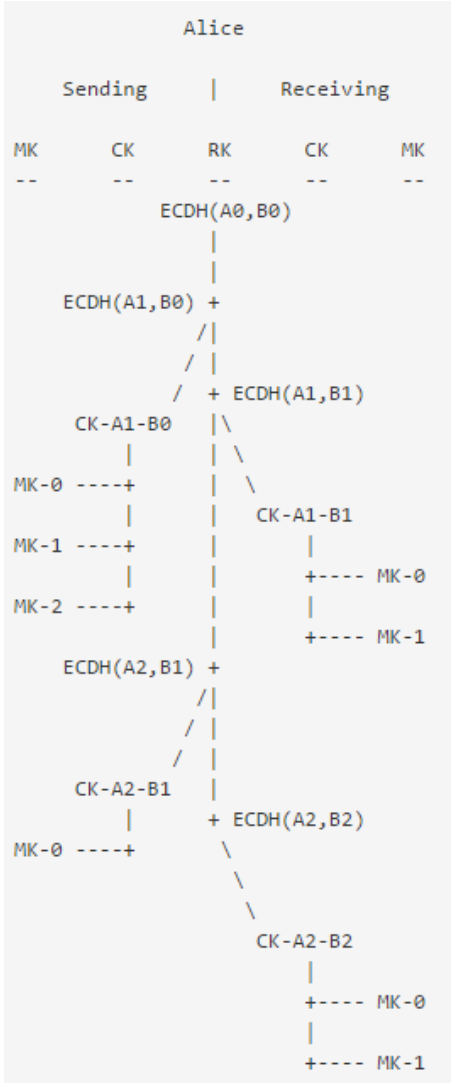
Introduction

Popular messaging apps switched to end-to-end encryption

- Great communication around it
- Privacy is now a requirement
- Debates at the government level to ask for backdoors: going dark ? Terrorists ?
- Increased feeling that those applications are unbreakable

Super crypto. But wait …

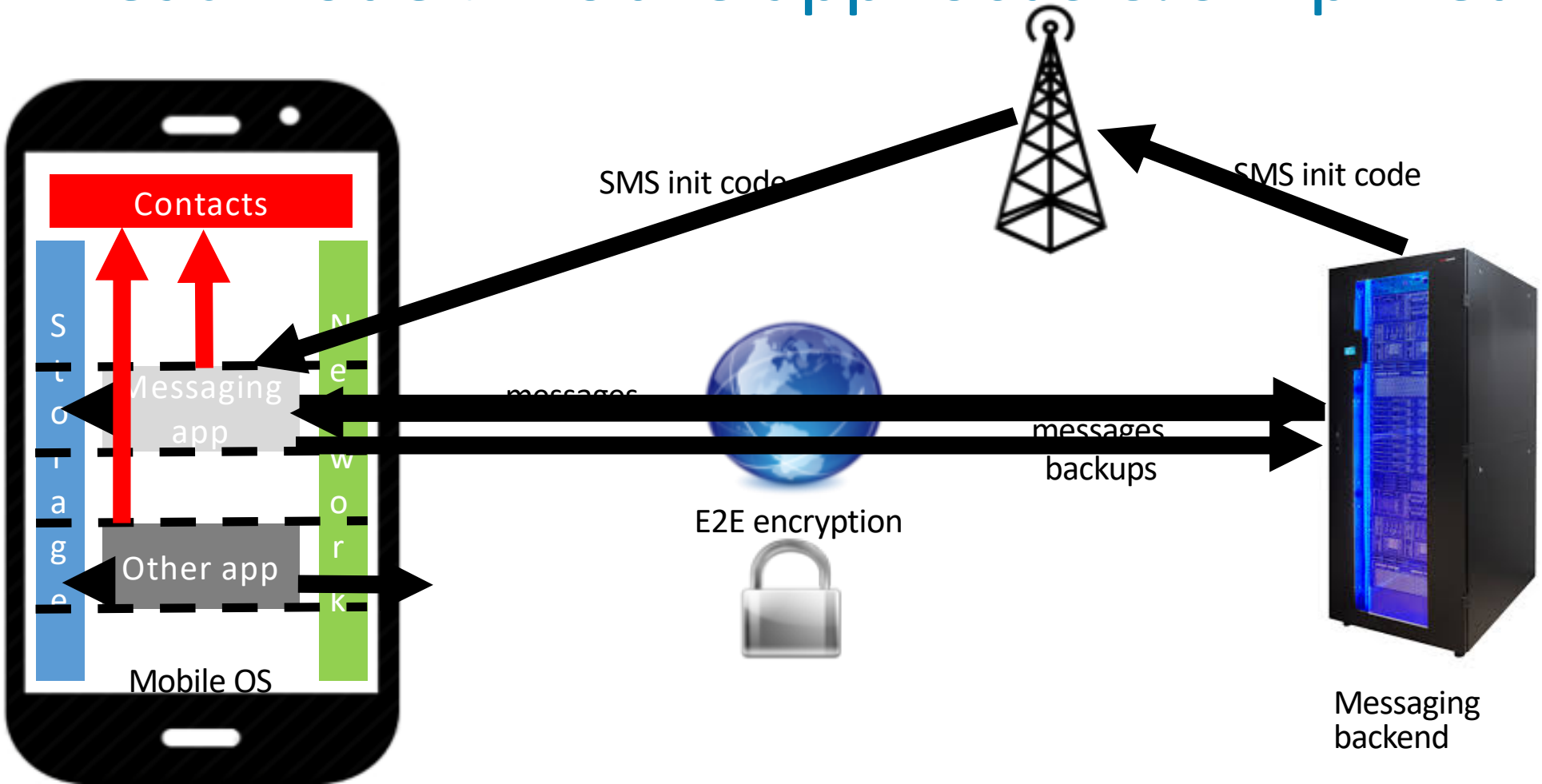
- Advanced ratcheting in Signal protocol: looks like an obvious flaw won't be here ➔
- But how messaging apps authenticate myself ?
  - No explicit identifier
  - Provisioning done by SMS
  - Link to device/phone number
- And my contacts ? Get them automatically from my address book



Man in the contacts

Laureline DAVID / Jérémy MATOS

Threat model: mobile app focus & simplified



Man in the contacts

Laureline DAVID / Jérémy MATOS

Accessing contacts

Easy to read/modify/create contacts

- There is an API for that
- Android example ➔

```
private boolean updateContactName(String phone, String newName) {
    ArrayList<ContentProviderOperation> ops = new ArrayList<ContentProviderOperation>();

    ops.add(ContentProviderOperation.newUpdate(ContactsContract.Data.CONTENT_URI)
            .withSelection(ContactsContract.CommonDataKinds.Phone.NUMBER + "=?", new String[]{String.valueOf(phone)})
            .withValue(ContactsContract.CommonDataKinds.StructuredName.DISPLAY_NAME, newName)
            .build());
    try {
        getContentResolver().applyBatch(ContactsContract.AUTHORITY, ops);
        return true;
    } catch (Exception e) {
        Log.e("oups","aie",e);
    }
    return false;
}
```

Shared data structure accessible in read/write

- Only restricted by permissions ➔
- And it contains authentication data in clear

```
<uses-permission android:name="android.permission.READ_CONTACTS" />
<uses-permission android:name="android.permission.WRITE_CONTACTS" />
```

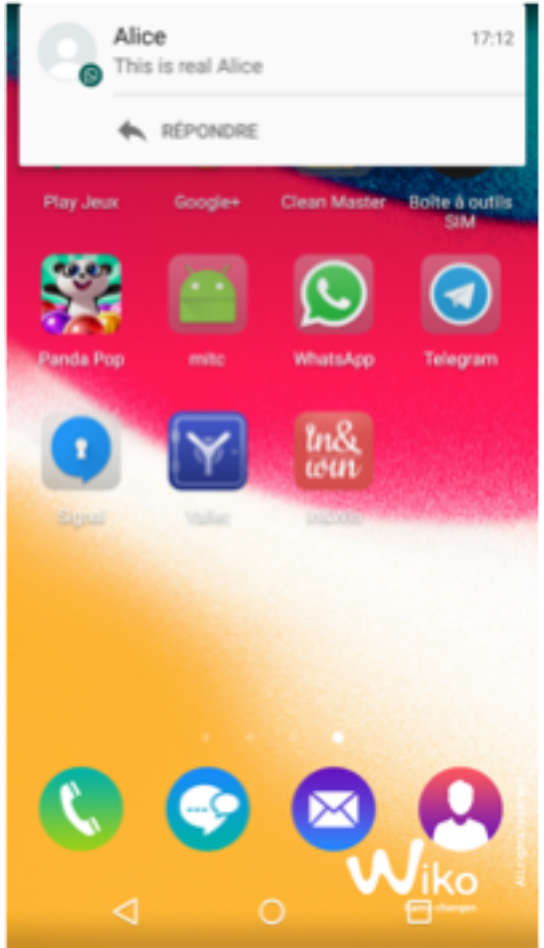
There is room for a **side channel attack : Man in the contacts**

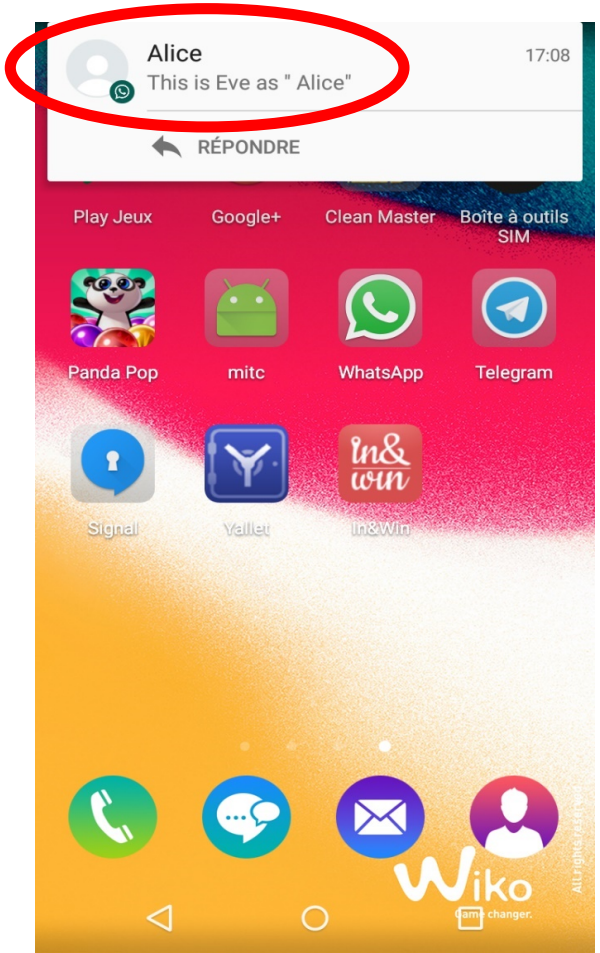
- Does not require a rooted device

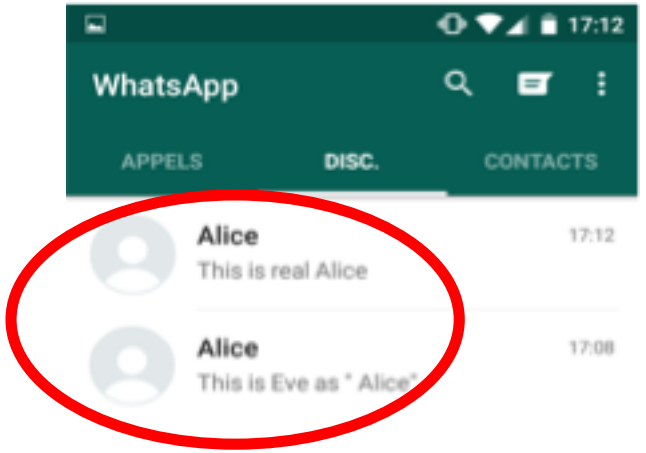
Man in the contacts

Laureline DAVID / Jérémy MATOS

Let’s create a new contact called “ Alice”



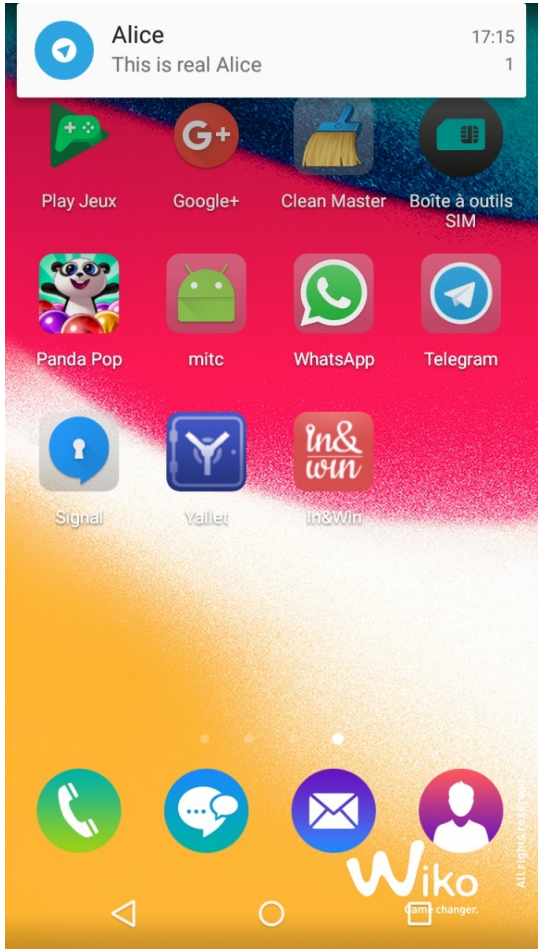


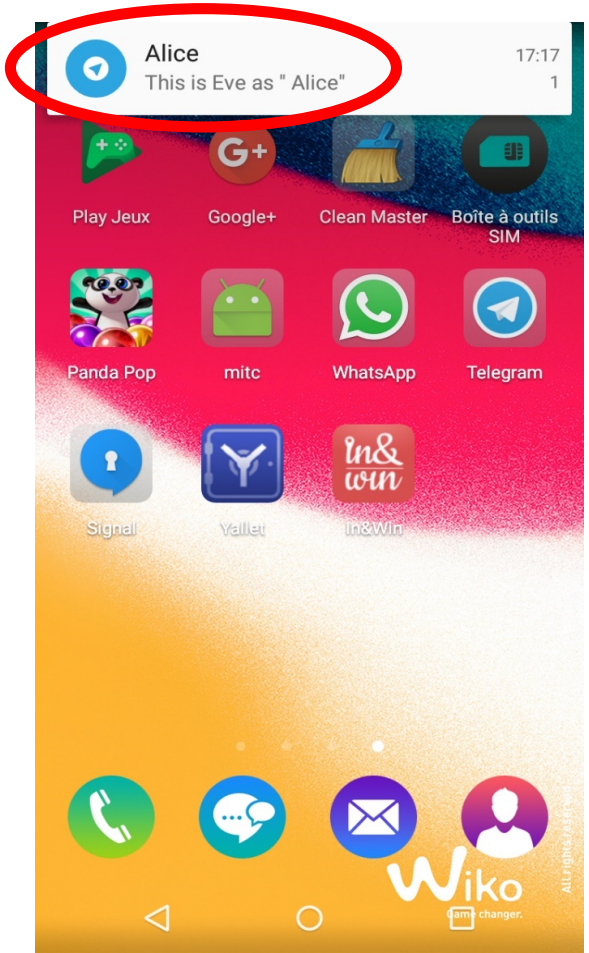


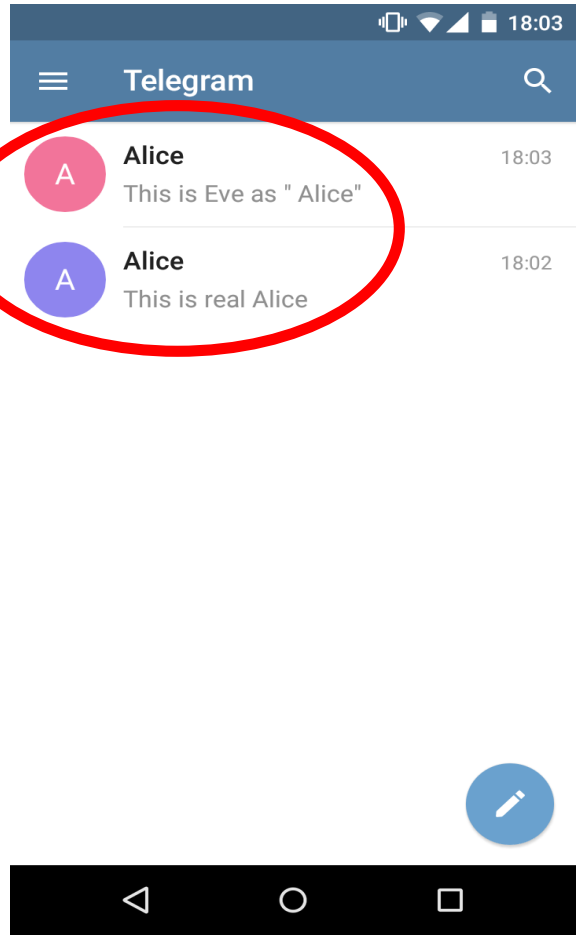
Man in the contacts

Laureline DAVID / Jérémy MATOS

Let’s create a new contact called “ Alice”



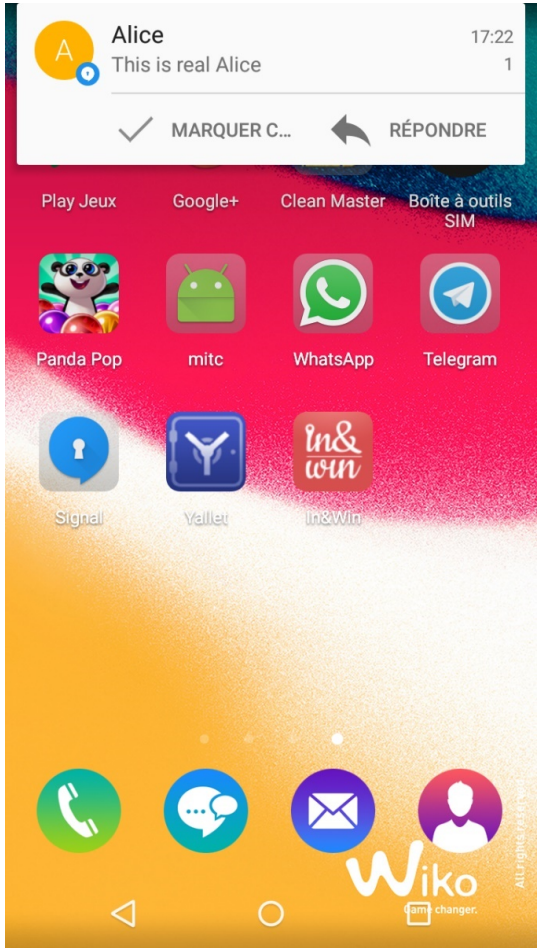


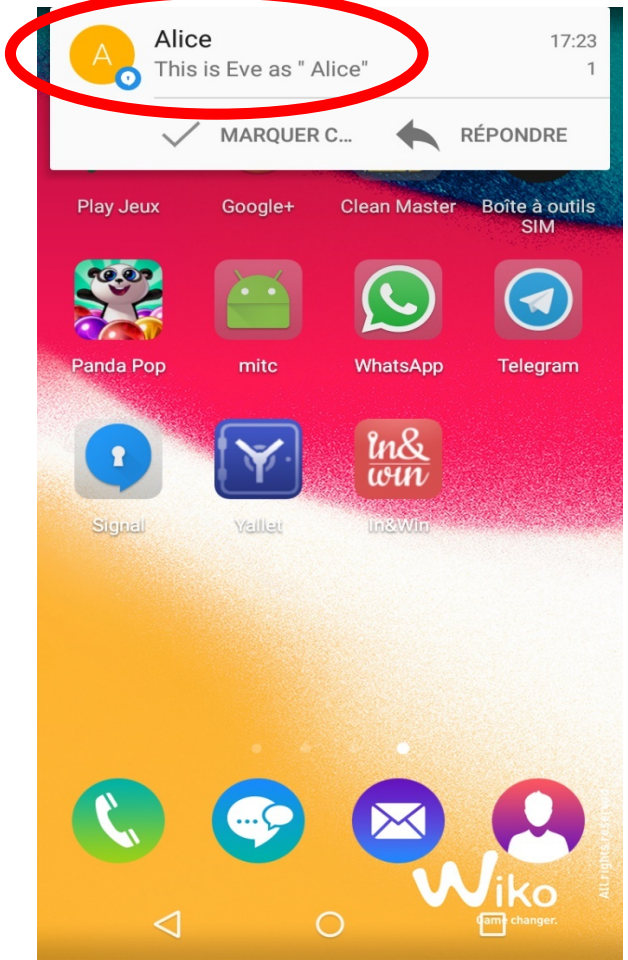


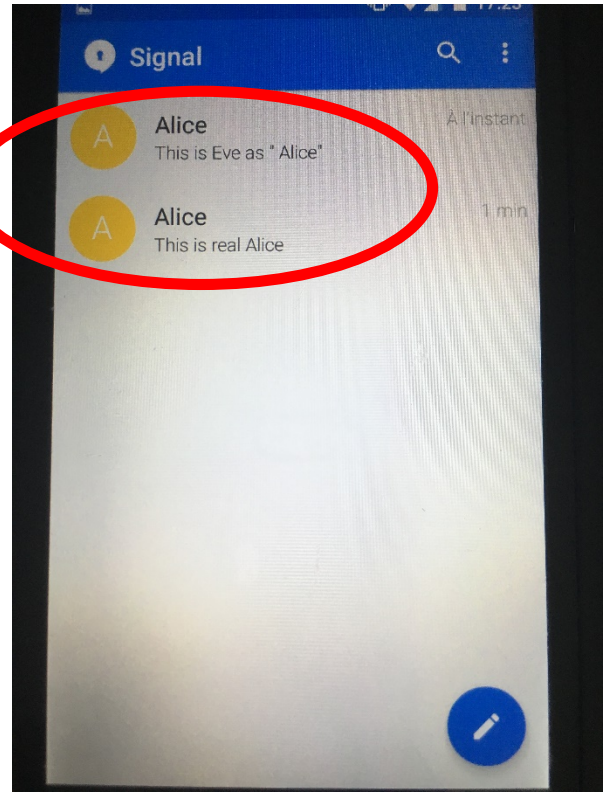
Man in the contacts

Laureline DAVID / Jérémy MATOS

Let’s create a new contact called “ Alice”







Man in the contacts

Laureline DAVID / Jérémy MATOS

Why does it work ?

- **Design error** from a security point of view: phone number as implicit identifier is a poor choice
- **Abusing Trust On First Use** (TOFU): new contact = new key = accepted by default
- **Same old trick of invisible characters**
- **End user/mobile not really included in the threat model**
  - Focus on protecting network/backend (e.g. from government agencies)
  - Side channel attack with some social engineering out of scope
  - E.g. Formal security analysis of Signal protocol: https://eprint.iacr.org/2016/1013.pdf

  Signal specifies a mandatory method for participants to verify each other's identity keys through an out-of-band channel, but most implementations do not require such verification to take place before messaging can occur

- **Bonus:** messengers store their own identifiers in the contacts

  ➔ we can know from another application if WhatsApp, Telegram or Signal is installed despite sandboxing

Man in the contacts

Laureline DAVID / Jérémy MATOS

Risk assessment

- Simple evaluation: risk = easiness of attack * user impact
- **Difficulty of attack: Low to Medium**
  - Technical: Low
    - Easy to access contacts via code
    - Not a problem to get MITC application approved for publication
  - Logistics: Medium
    - One phone number is enough
    - But need to convince many users to install the application
- **User impact: High**
  - Thousands of users can be spied via those 3 famous messengers

Difficulty to attack	Low business impact	Medium business impact	High business impact
Low	Low	Medium	Very High
Medium	Low	Medium	High
High	Low	Low	Medium

Man in the contacts

Laureline DAVID / Jérémy MATOS

Vendors feedback

Telegram

security@telegram.org = /dev/null

WhatsApp (Facebook)

*We appreciate your report. **Ultimately** an attacker with **malware** installed on a device is going to be able to alter data on the device itself. In your examples for **WhatsApp conversations remained properly bound to the phone number that the messages were sent to**. Beyond that, WhatsApp allows people to set local aliases for contacts and to view the number associated with a specific message thread at any point. Given that, we don't feel that this behavior poses a significant risk and **we do not plan to make any changes here**. Please **let us know if you feel we've misunderstood something here**!*

Signal (Moxie Marlinspike)

*Hey Jeremy, saw your support email about "man in the contacts." This, like all interception techniques, is **what safety numbers are for. Signal users would be notified that the safety numbers for their contact have changed**, and be asked to verify them. A successful MITM attack would need to find a way to intercept communication without triggering that notice.*

---

*Hey Jeremy, **Signal is not designed to protect your device against malware**. Thanks for getting in touch, good luck with everything.*

Man in the contacts

Laureline DAVID / Jérémy MATOS

Implementing Man in the contacts in practice

Bachelor thesis of Laureline

1. Android game: a social version of Rock, Paper, Scissors

- Available on Google PlayStore at https://play.google.com/store/apps/details?id=com.tricktrap.rps
- Approved without any issue on July 2017

  Biggest challenge: contact API is a total mess

2. The Command-and-Control server

- Python based
- Overview of its web interface

Man in the contacts

Laureline DAVID / Jérémy MATOS

Spear-phishing in action

- Convince Alice to download the social game
- When Alice starts game, C&C server controlled by Eve is notified that a secure messenger is installed and now listens for contact orders.
- Eve asks C&C server to create “ Bob” in Alice’s phone.
- Eve send a malicious link to Alice as “ Bob” via secure messenger.

  Alice clicks on it because she knows well Bob and it is a super secure messenger.
- **Backend servers cannot filter dangerous content because by design they can’t read the messages**.

- NB: Desktop versions of those secure messengers are more and more popular

Live demonstration

- **Any volunteer ?**

Man in the contacts

Laureline DAVID / Jérémy MATOS

Open sourcing the tool

Source code of Android game available at https://github.com/ltouroumov/rockpaperspam-client

Source code of Command-and-Control server is available at https://github.com/ltouroumov/rockpaperspam-server

Don’t hesitate to share those links

Vendors may now better understand that it can be a real-life issue.

Man in the contacts

Laureline DAVID / Jérémy MATOS

Next attack to implement: Man in the Middle

- Convince Alice and Bob to download the social game
- When Alice and Bob start the game, Eve notices they have a secure messenger installed
- Eve asks C&C server to create “ Alice” in Bob’s phone and “ Bob” in Alice’s phone
- Eve sends as “ Alice” via secure messenger an introduction message to Bob.

  Bob is thinking he is talking to real Alice.
- Eve forwards as “ Bob” the answer of Bob to Alice.

  Alice is convinced it is the real Bob because it’s really his words.
- Alice answers to ” Bob”, who is in fact Eve. Eve can now forward the message to Bob as “ Alice”.

  Bob is also convinced. ➔ We now have a full Man in the Middle

- **Alice and Bob are using E2E encrypted communications but they are talking to the wrong contact without noticing**

Possible mitigations

End users

- Avoid installing applications requesting contact permissions.

  But this is the case for secure messengers ….

Mobile OS

- Sandbox contact information
- Be stricter on write operations to address book

Secure messengers

- Give up implicit trust on contact information:

  require users to manually add people they are talking to
- Or **raise user awareness when a conversation is starting with a brand new contact**:

  make it clear in U.I. this an unusual situation, e.g. with a danger sign

Man in the contacts

Laureline DAVID / Jérémy MATOS

Conclusion

E2E encryption cannot guarantee privacy if you're not sure who you are talking to

- Beware of messages showing good cryptography is used because it can bring a false sense of security

Security model around contacts if far too open for sensitive apps

- If the design of your solution includes access to contacts, start with a threat modeling session

Tool open sourced to demonstrate secure messengers should not be trusted blindly

- Their great security reputation can be used against them for a successful social engineering attack
- By design dangerous messages cannot be filtered