

Rainbow Crypt: Securing Communication through a Protected Visual Channel

Alberto Bartoli, Giorgio Davanzo, Eric Medvet
DI³ — Università di Trieste, Italy

<http://bartoli.inginf.units.it>

November 2011

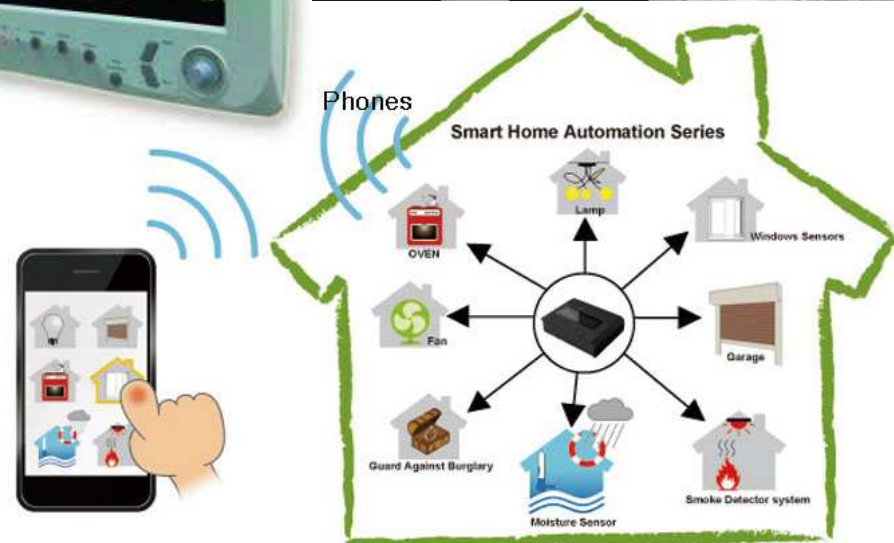


Scenario

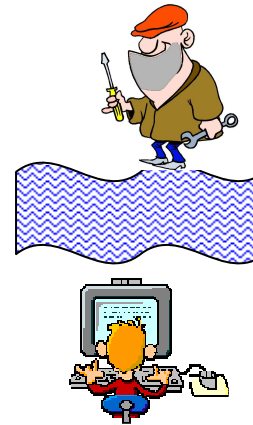


- Environment with “a lot” of wireless-enabled devices, sensor, gadgets

- Patient / hospital monitoring
- Home automation
- Access control
- ...
- “Ambient Intelligence”
- “Internet of Things”
- ...



Problem



- ❑ Ensure Mutual Authentication of devices (**Secure Pairing**)
 - ❑ Device A should communicate with device B
 - ❑ (Possibly hostile) Device X must not be able to impersonate either device to the other

- ❑ **Very difficult:**
 - ❑ Wireless medium ⇒ Messages may be **intercepted** and **replayed** easily
 - ❑ Hostile devices may be easily **concealed**

Theory: Simple

Practice: Very difficult



□ Theory

- Establish secret key between devices (somehow)
- Plenty of protocols ensuring mutual authentication and secrecy

□ Practice

- VERY difficult to deploy these protocols
- They rely on assumptions not met in this scenario
- Their use is often difficult for the unexperienced user

Example (basic idea)

- ❑ User identifies devices A and B “visually” (with his hands) and wants a secure pairing
- ❑ User must tell A that its partner is B
 - ❑ A must have a **suitable input channel** (e.g. a keyboard)
- ❑ How to identify B ?
 - ❑ B must have a **suitable output channel** (e.g. an LCD screen)
 - ❑ B must be certificated by a **trusted authority**, or
 - ❑ None of the above holds in practice

Real-world example: Wi-Fi “Protected Setup” (I)

- ❑ Part of Wi-Fi specification
- ❑ Enables “easy and secure setup” between pairs of devices (**secure pairing**—sort of)
- ❑ Implemented in thousands of products



- ❑ **PIN-based**
 - ❑ The device must support the entering of a password
 - ❑ Inconvenient or unfeasible for many kinds of devices
- ❑ ...

Real-world example: Wi-Fi “Protected Setup” (II-a)

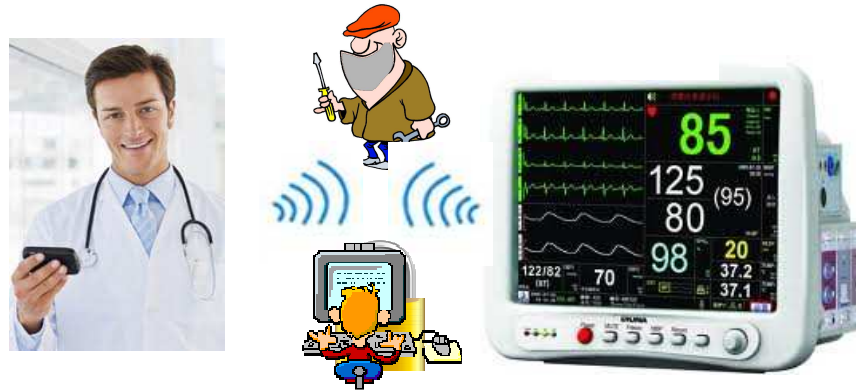
❑ Push Button Configuration (PBC)

- ❑ The user pushes a (possibly virtual) button on the devices more or less at the same time
- ❑ The devices establish a shared secret key



Real-world example: Wi-Fi “Protected Setup” (II-b)

- ❑ The problem is, it does **not** ensure secure pairing !



- ❑ “Users should be aware that **during the two-minute setup** period which follows the push of the button, **unintended devices could join the network** if they are in range.”

Wi-Fi Protected Setup FAQ @ Wi-Fi Alliance

Proposed solutions

❑ Out-of-band channel

- ❑ An **additional** channel is assumed to exist
 - ❑ ...and this channel is **secure** by definition
- ❑ Shared secrets are established on this channel
 - ❑ Devices must have I/O capabilities for this channel and for the user
- ❑ *Several examples: text-to-speech, displays and alike*

❑ Novel Wi-Fi protocols

- ❑ Recent proposal by MIT researchers
USENIX Security Symposium 2011

A very simple solution



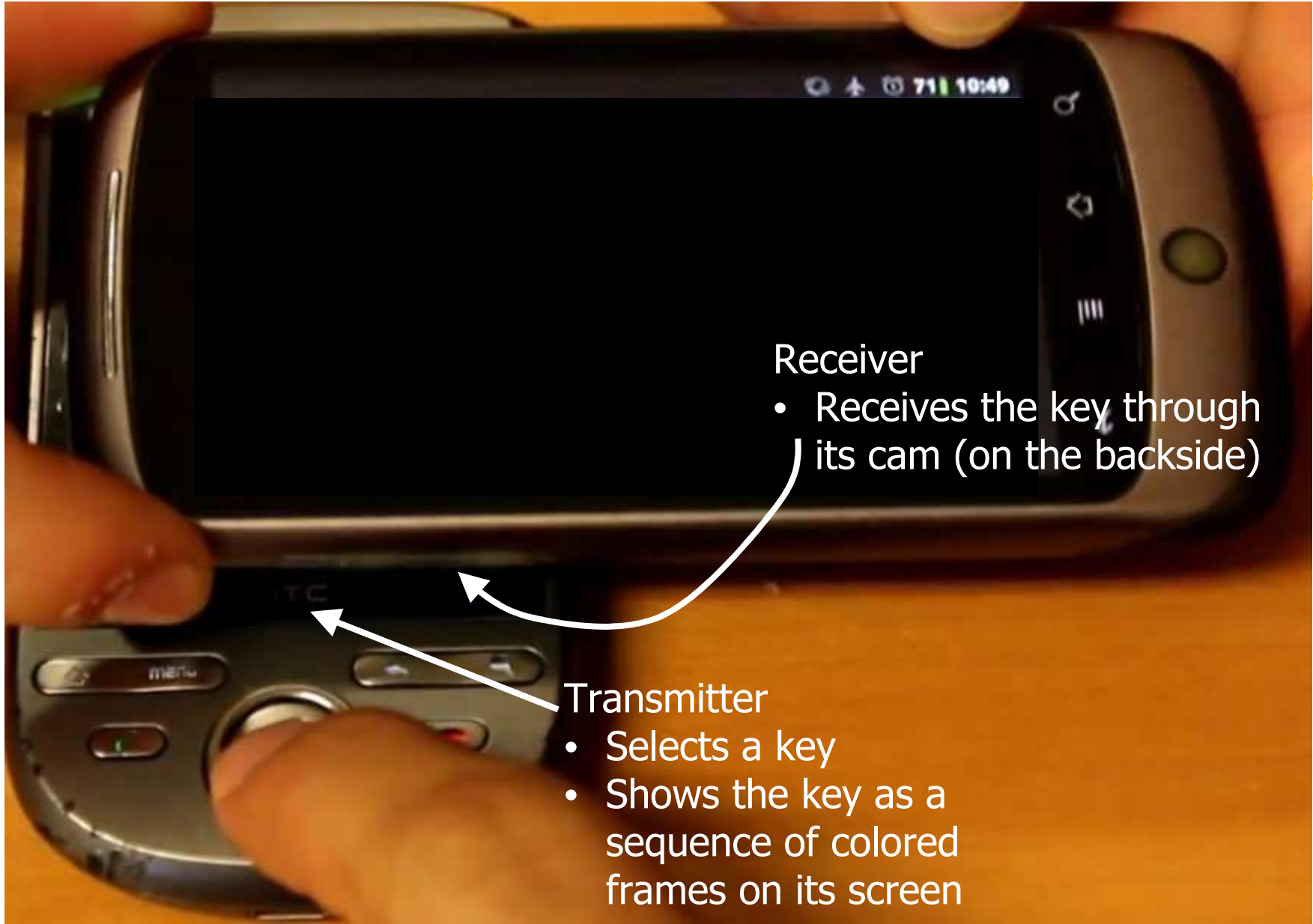
- ❑ **Physical contact** (*Stajano and Anderson 2000*)
 - ❑ User places devices A and B in physical contact (electrical)
 - ❑ While in contact, devices establish a shared secret
- ❑ **Systematic, Simple, Effective:**
 - ❑ User selects exactly the devices he needs (no concealed devices)
 - ❑ Attacker cannot intercept anything (no communication attacks)

- ❑ Did not succeed due to the **lack of common interfaces**

Our work



- ❑ We demonstrate the Stajano&Anderson approach
 - ❑ May indeed be applied in practice
 - ❑ For a very specific, but very important, class of devices
- ❑ **Smartphones** (and potentially many other devices)
 - ❑ Small LCD color screen on one device
 - ❑ Low-end color camera on the other device
- ❑ We place the two devices **next to each other**
- ❑ A **few seconds** suffice to establish “reasonably long” keys
(≈ 10 bps; 42 bits in 5 sec)
- ❑ Keys **cannot be intercepted**
- ❑ Quite more difficult than it seems. An empirical work



Receiver

- Receives the key through its cam (on the backside)

Transmitter

- Selects a key
- Shows the key as a sequence of colored frames on its screen

Short demo app (video)



- Transmitter
 - Sends key K **and** text string encrypted in K
- Receiver
 - **Displays** on its screen what is being received (obviously it should not)

http://www.youtube.com/watch?v=FHPa_hx1R1M

Practical problems:

Focus distance



- Fact
 - Cameras of our interest require at least 50 cm from the subject
 - We use them instead at near-zero distance

- Key consequence
 - Every received frame is going to be a **single** blurred color

- Solution (fundamental constraint)
 - Each transmitted frame must be a uniform color image

Practical problems: White balancing



- Fact
 - White balancing camera software attempts to average the image color to a neutral gray

- Key consequence
 - When the transmitted frame is a uniform color image, the received frame is a uniform but **different** color
 - Result hardly predictable: depends on settings and type of the device

- Solution
 - Disable white balancing software :-)

Practical problems:

Exposure



- Fact
 - Camera software modifies exposure interval of the sensor to obtain a standard luminosity

- Key consequence
 - The transmitted uniform color tonality may be received as an entirely **different** uniform color tonality

- Solution (...no, cannot be disabled in software...)
 - Quite a few experiments to figure out **how many different colors** can be detected reliably... and **which ones**

Symbol alphabet

- As it turns out:
 - Only **4 colors** may be discerned (2 bits / frame)
 - It does not matter which ones, as long as they are “sufficiently different” from each other

- We used

□ Red	255	0	0
□ Blue	0	0	255
□ Violet	255	0	255
□ Black	0	0	0

- Receiver: algorithm $YUV \Rightarrow RGB$

Transmission rate

- ❑ Fact:
 - ❑ Maximum frame sampling rate depends on the hw
 - ❑ Our cheapest platform supported 21 fps (42 bps)
- ❑ Early experiments
 - ❑ Transmission rate 10 fps (20 bps)
- ❑ Very bad results



Practical Problem: Platform Limitations

- Fact
 - Android API allows applications to specify the desired frame sampling rate

 - But:
 - frame rate is **not uniform**
 - there may be **significant variance**
 - desired rate guaranteed only over “long periods”

Example



Transmitted
(5 fps)



Received
(20 fps)
IDEAL



One
in excess

One
in excess

One
missing

Two
in excess

REAL

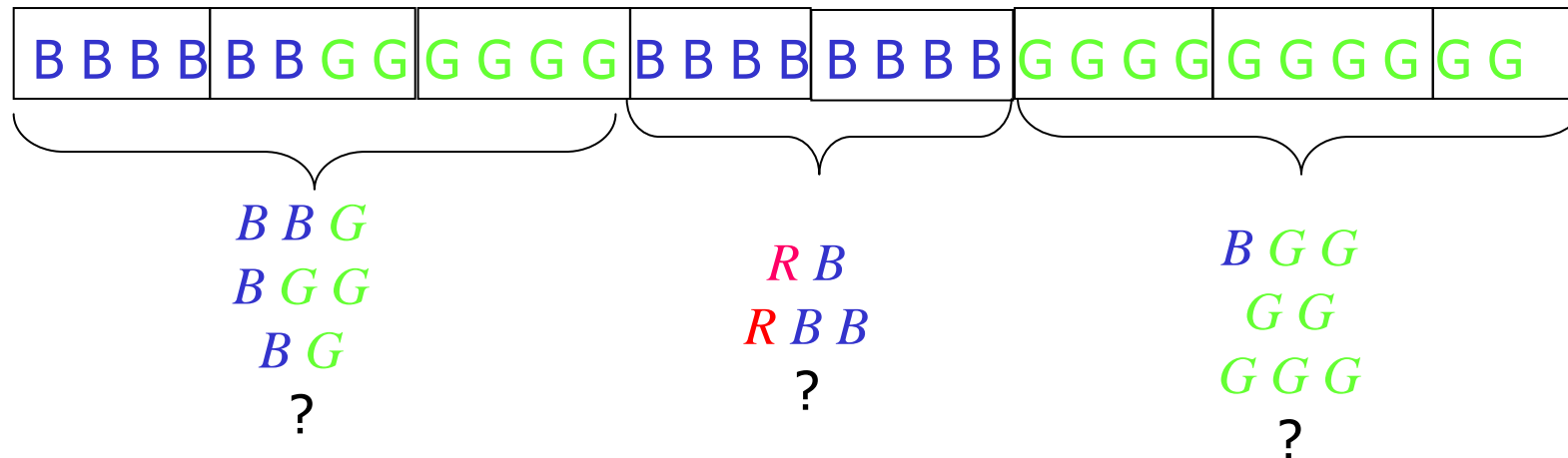
Obvious consequence: Out of sync

B B B B G G G G R R R R B B B B

Sequence of received frames

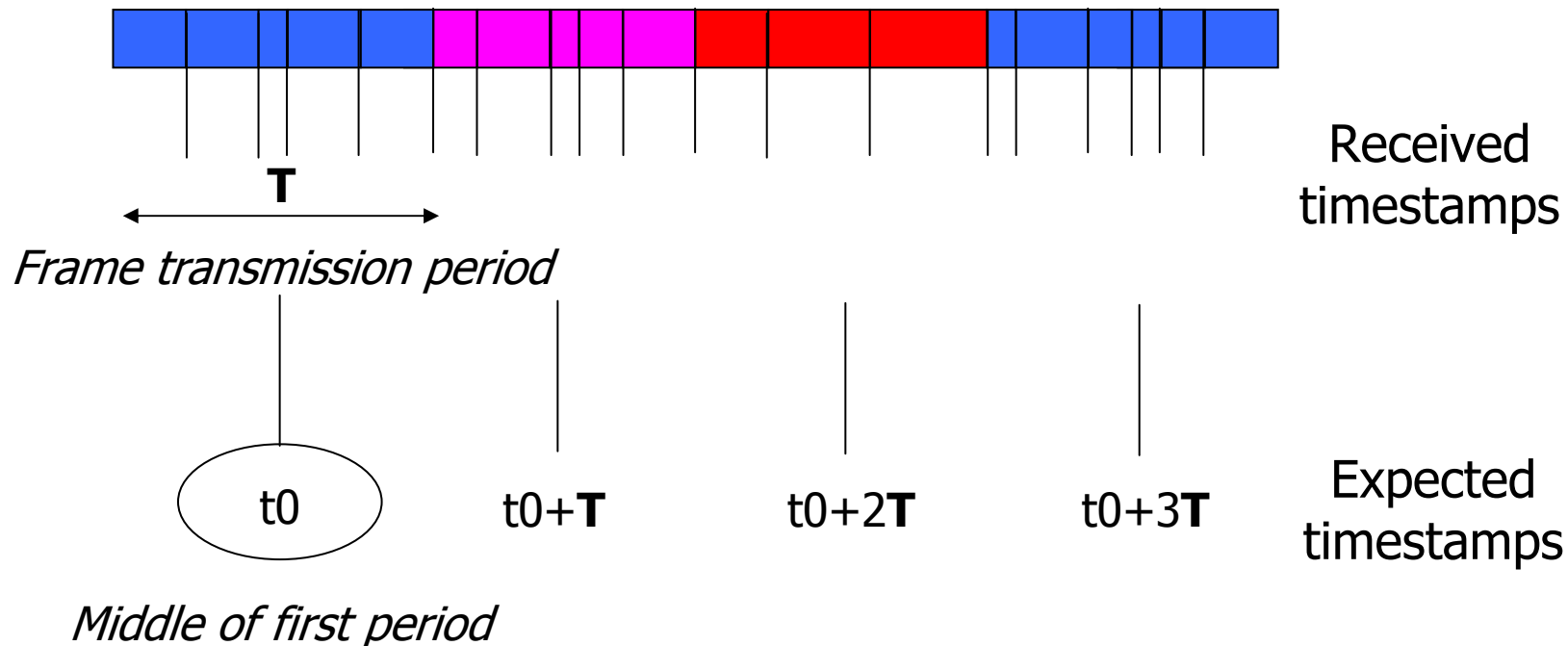
B *G* *R* *B*

Chosen symbol



Our solution (I)

- Upon receiving a frame $F \Rightarrow$ Associate F with timestamp $t(F)$



- Methods for

- Choosing F at each **expectedTimestamp**
- Based on sequence of $\langle F_i \text{ — receivedTimestamp}_i \rangle$

Our solution (II)

- The K-th symbol is selected as follows:
- **Nearest**
 - Frame with timestamp closer to $t_0 + KT$
- **Mean**
 - Associate each frame with closest expected timestamp
 - Average all frames in each group
- **Weighed Mean**
 - Like Mean,
with weights depending on squared distance from $t_0 + KT$
- **Weighed Mean with Sync**
 - Like Weighed Mean,
with initial **RBRB** sync sequence

Solution (III)

- ❑ Quite a few exploratory experiments for choosing:
 - ❑ Transmission rate: 5 fps (**10 bit/s**)
 - ❑ Sampling rate: 20 fps

Method	Average (%)	StdDev (%)
Nearest	12.0	23.6
Mean	11.4	20.2
wMean	10.3	20.2
wMean Sync	0.3	1.0

Concluding remarks

- ❑ Secure pairing for an important class of devices
 - ❑ Several (potential) interesting applications
 - ❑ Systematic, Simple, Effective—Stajano&Anderson

- ❑ Pairing channel 10 bps
- ❑ Severely limited, but suffices for initial key exchange (42-bits in ≈ 5 sec)

- ❑ Certainly many opportunities for improvement (transmission method quite naive)
- ❑ Hardware will certainly improve



**Thanks for your
attention...**



Practical Problem: Exposure (again)

- ❑ Transmitted symbol is “too dark” .
- ❑ Receiver increases exposure:
 - ❑ Sampling interval much greater than 50 ms
 - ❑ May even be greater than 1 s (with black frames)
- ❑ Transmitter sends K symbols
- ❑ Receiver merges them all into one symbol
- ❑ Not quite predictable
- ❑ Not controllable in software / Cannot be disabled



Practical Problem: Exposure (again)

- Transmitted symbol is “too dark” \Rightarrow
- Receiver increases exposure:
 - Sampling interval much greater than 50 ms
 - May even be greater than 1 s (with black frames)
- Transmitter sends K symbols
- Receiver merges them all into one symbol
- Not quite predictable
- Not controllable in software / Cannot be disabled

