Technology Transfer from Security Research Projects

A Personal Perspective

N. Asokan
Aalto University & University of Helsinki
http://asokan.org/asokan/research
Five examples

• Optimistic Fair Exchange
• Generic Authentication Architecture
• Channel Binding in Protocol Composition
• Secure Device Pairing
• On-board Credentials
Five examples

• Optimistic Fair Exchange
• Generic Authentication Architecture
• Channel Binding in Protocol Composition
• Secure Device Pairing
• On-board Credentials
Fair Exchange

How can two mutually distrusting parties exchange digital “items” on the Internet?

Existing solutions:

Gradual Exchange protocols

Trusted Third Party protocols
Fair Exchange: design choices

• Common case: both *want to* complete the exchange
  – design protocol that is efficient for the common case
  – but allows recovery in case of exceptions

• Requirements
  – Effectiveness
  – Fairness
  – Timeliness
  – (Non-invasive)
Optimistic Fair Exchange

Alice | Bob

Resolve

http://www.semper.org/
Optimistic Fair Exchange: Recovery

Resolve

Alice

if A-item matches B-exp
• extract B-item from B-permit
• store A-item

extract

B-exp

B-item

B-permit
Optimistic Fair Exchange

Alice Bob

http://www.semper.org/
Optimistic Fair Exchange: Recovery

**Abort**

If not resolved, issue abort token

**Resolve**

If not aborted, and if A-item matches B-exp
- extract B-item from B-permit
- store A-item

Resolve for Bob is similar
Verifiable Encryption

Analogy - jewelry in a glass box: can see but can’t touch
Verifiable Encryption of discrete logs

Setting: secret = s ∈ G1, desc d = g^s (in G2)

Prover

s0 ∈R G1, v ← g^{s0}
s1 ← s0 – s
Ei ← Enc(ri, si), i={0,1}

Verifier

v, E0, E1

b ∈R {0,1}

rb, sb

(\text{verifyEnc})

Repeat n times
(cut-and-choose)

TTP

Eb

\tilde{s}b ← Dec(Eb)

s ← sb + \tilde{s}b

recover
From Verifiable Encryptions to Permits

\[ \text{A-exp} = \text{desc. of } \text{B-item} \]

\[ \text{A-permit} = \text{Verifiable Encryption of } \text{A-exp} + \text{A-item} \]

Optimistic Fair Exchange: the aftermath

• Someone has to run the Third Party
  – Wants to monetize every transaction!
Verifiable Encryption of discrete logs

Setting: secret = s ∈ G1, desc d = gs (in G2)

Prover

s0 ∈R G1, v ← gs0
s1 ← s0 − s
Ei ← Enc(ri, si), i={0,1}
v, E0, E1

Verifier

b ∈R {0,1}
rb, sb

Repeat n times (cut-and-choose)

Verifier

(Eb) b

TTP

s0 ← Dec(Eb)
sb

s ← sb + sb

verifyEnc

recover
Verifiable Encryption of discrete logs

Setting: secret = s ∈ G1, desc d = gs (in G2)

\( s0 \leftarrow s0 - s \)
\( v, E0, Cert \)
\( s1 \)
\( (d \cdot g^{s1} = v?) \land \land \text{verify}(Cert) \)

Pre-paid coupons bought from the TTP to be used for every optimistic transaction!
Optimistic Fair Exchange: the aftermath

• Someone has to run the Third Party
  – Wants to monetize *every* transaction!

• 16 years on, current status:
  – Reputation systems
  – In-line TTP (e.g., E-bay escrow service)
Continuing “impact” in research circles!

Autumn 2015
Continuing “impact” in research circles!

Nov 2, 2016
Optimistic Fair Exchange: the aftermath

• Someone has to run the Third Party
  – Wants to monetize every transaction!

• 15 years on, current status:
  – Reputation systems
  – In-line TTP (e.g., E-bay escrow service)

• Impact in academia vs. real world impact

• Biggest impact of SEMPER?
  http://logging.apache.org/log4j/2.x/
Optimistic Fair Exchange: lessons learned

- Don’t just guess security requirements; Ask stakeholders
- Desiderata for deployment and research can be different
  - “the more (independent) parties you require for your scheme, the less likely it will be deployed”
- Capturing researcher interest (Tech transfer) Impact
  - MANETs anyone?
- “90-10 rule” applies to deploying security
  - “Good enough beats perfect”
Five examples

• Optimistic Fair Exchange
• Generic Authentication Architecture
• Channel Binding in Protocol Composition
• Secure Device Pairing
• On-board Credentials
Generic Authentication Architecture

Can we bootstrap a general-purpose global-scale authentication and authorization infrastructure from the existing cellular security infrastructure?

- Need was evident:
  - “Global PKIs will not happen”
- Ad-hoc bootstrapping already in use
  - e.g., Coke vending machine accepting payments via SMS, 1997
- Idea: Bootstrap short-lived certificates from “local PKIs”
Bootstrapping a "local PKI"

Global Cellular Authentication/authorization Infrastructure

Home Security Server

Authentication & Key Agreement (AKA)

IK, CK

Serving Network

RA

CA

SP

IK, CK

PK_D/SK_D

Cert_D

K

IK, CK

K
3GPP “Generic Authentication Architecture”

Two layer architecture
- Generic Bootstrapping Architecture (GBA)
- Specialized Application Servers
  - E.g., for “subscriber certificates”

HSS

Credential Fetching Protocol

Bootstrapping Server

Key distribution Protocol

Application Server

Bootstrapping Protocol

User Equipment (UE)

Bootstrapping client

Application client

Application Protocol

Relevant 3GPP documents: E.g., [33.919], [33.220]
GAA: the aftermath

• Standardized in 3GPP
  – Variants: GBA and GBA_U (implemented in the smartcard, UICC)
  – GBA implemented for some services
  – none of which has taken off (e.g., Mobile TV)
    • At least not yet!

• Today’s solutions:
  – Bootstrapping: Facebook, Google, …
    • Some mobile carriers even deployed PKI-enabled SIM cards
  – Roaming: iPass, Shibboleth, …

• Variants of the idea had more success
  – E.g., EAP SIM
GAA: lessons learned

- (Standardization) Politics can suffocate a good idea
- “90-10 rule” applies to deploying security
Five examples

- Optimistic Fair Exchange
- Generic Authentication Architecture
- Channel Binding in Protocol Composition
- Secure Device Pairing
- On-board Credentials
Channel Binding in protocol composition

Composing two secure authentication protocols carelessly can lead to a man-in-the-middle vulnerability

- Protocol composition can ease deployment
- Examples:
  - Server auth. using TLS + user auth. with password
  - Authentication for VPN access using legacy credentials
  - Bootstrapping a "local PKI"
3G AKA

Provides mutual authentication
Bootstrapping certificate enrollment

1. Set up a (server-authenticated) TLS channel

   1.1. Serving Network RA
   1.2. Home Security Server

2. Run AKA

   2.1. IMSI
   2.2. RAND, AUTN
   2.3. RES
   2.4. STOP if RES ≠ XRES

3. Do certificate enrollment via the (mutually) authenticated TLS channel

   3.1. Cert Request
   3.2. Cert Response

   STOP if SQN ≤ SQNu

RAND, AUTN, XRES, IK, CK
Bootstrapping certificate enrollment

1. Set up a (server-authenticated) TLS channel

2. Run AKA

3. Do certificate enrollment via the (mutually) authenticated TLS channel

Channel binding: Use of **cryptographic binding** to compose two authenticated channels

Channel binding: the aftermath

• Fiery reception at Security Protocols workshop!
  – “But you are using the worst rackets in industry as a justification for what you’re doing. There are all sorts of people just generating garbage protocols, a couple of which you have already mentioned here. We’re trying to reverse their work, whereas you’re trying to advocate we use all these garbage protocols.”
  – For an entertaining read, see transcript of discussion during my talk at SPW ’03!

• Impact in IETF
  – Closing down of ipsra working group; channel binding in IKEv2
  – Continued attention: e.g., RFC 6813
Channel Binding: lessons learned

- Negative results are useful for security practitioners
- Standardization can make a good idea see light of day
- (Tech transfer) Impact $\rightarrow$ Capturing researcher interest
Five examples

• Optimistic Fair Exchange
• Generic Authentication Architecture
• Channel Binding in Protocol Composition
• Secure Device Pairing
• On-board Credentials
Secure Device Pairing

How can the process of pairing two devices be made easy to use without compromising security or adding to cost?
Secure Device Pairing: ca. 2005

Cracking the Bluetooth PIN*

Yaniv Shaked and Avishai Wool
School of Electrical Engineering Systems,
Tel Aviv University, Ramat Aviv 69978, ISRAEL
shaked@eng.tau.ac.il, wool@eecs.tau.ac.il

Abstract
This paper describes the implementation of an attack on the Bluetooth security mechanism. Specifically, we show that
new primitives to be broken, because new cryptography
is less sound and may contain hidden flaws. Further-
more, Bluetooth is designed for short-range communi-
cation (around one meter of each other). This short range is

Security Weaknesses in Bluetooth

Markus Jakobsson and Susanne Weitzel
Lucent Technologies - Bell Labs
Information Sciences Research Center
Murray Hill, NJ 07974
USA
{markus,swetez}@research.bell-labs.com

Abstract. We point to three types of potential vul-
nerabilities in the Bluetooth standard, version 1.0B. The first vulnerability opens up the system to an attack in which an adversary under certain circumstances is able to determine the key exchanged by two wireless devices, making

*Cracking the Bluetooth PIN is a research topic, not a commercial product.
Naïve usability measures damage security

Pictures taken with mobile phone showed up on neighbour's TV

- Default password must be changed when starting to use Bluetooth-equipped devices; read the manual!

elsewhere as well. It is, therefore, absolutely essential that the password is changed immediately when the device is first installed."

"This is clearly printed in the user's manual", Rosenberg points out. How often have we heard that before?

"Once the digital receiver's password has been changed, the new password also has to be entered in the transmitting device, in this case, the TV. Otherwise, the receiver will not be able to send any further information."

http://www.helsinki-hs.net/news.asp?id=20030930IE16
Naïve security erodes usability

Car kits
- Allow hands-free phone usage in cars
- Retrieve/use session keys from phone SIM
  ➢ users must enter 16-character passcodes

More secure = Harder to use?

Cost:
Calls to Customer
Key establishment for secure pairing ~2005

Key establishment

Key transport via OOB channel

Key agreement

Symmetric crypto only

Authenticated

Unauthenticated

Asymmetric crypto

Authenticated

Unauthenticated

Short keys vulnerable to passive attackers

Secure against passive attackers
Authentication by comparing short strings

\[ v_A \leftarrow H(A, B, PK_A|PK'_B) \]
\[ v_B \leftarrow H(A, B, PK'_A|PK_B) \]

\( v_A \) and \( v_B \) are short strings (e.g., 4 digits),
User approves acceptance if \( v_A \) and \( v_B \) match
A man-in-the-middle can easily defeat this protocol
MitM in comparing short strings

Pick $PK_{C2}$ by trial-and-error:

$H(A, B, PK_A|PK_{C2}) = v'_B$

$v'_B \leftarrow H(A, B, PK_{C1}|PK_B)$

If $v'_B$ is $n$ digits, attacker needs at most $10^n$ guesses; Each guess costs one hash calculation

A typical modern PC can calculate 100000 MACs in 1 second
Authentication by comparing short strings

Choose long random $R_A$
Calculate commitment $h_A \leftarrow h(A, R_A)$

$h()$ is a hiding commitment; in practice SHA-256

User approves acceptance if $v_A$ and $v_B$ match

$2^{-l}$ ("unconditional") security against man-in-the-middle ($l$ is the length of $v_A$ and $v_B$)

[LAN05] MANA IV, IACR report; [LN06] CANS '06
## Key establishment for secure pairing ~2008

<table>
<thead>
<tr>
<th></th>
<th>Unauthenticated Diffie-Hellman</th>
<th>Authenticated Diffie-Hellman</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>short-string comparison</td>
<td>short PIN</td>
</tr>
<tr>
<td>WiFi Protected Setup</td>
<td>“Push-button”</td>
<td>√</td>
</tr>
<tr>
<td>Bluetooth 2.1</td>
<td>“Just-works”</td>
<td>√</td>
</tr>
<tr>
<td>Wireless USB</td>
<td>√</td>
<td></td>
</tr>
</tbody>
</table>

[AN10] “Security associations for wireless devices” (Overview, book chapter)

Secure Pairing: the aftermath

- Widely deployed (Bluetooth SSP, WiFi Protected Setup)
- Improving usability/security → fundamental protocol changes

[UKA07] "Usability Analysis of Secure Pairing Methods", USEC ‘07
Secure Device Pairing: lessons learned

- Address pain points - builds credibility with stakeholders
- Don’t just guess security requirements; Ask stakeholders
- Desiderata for deployment and research can be different
- Standardization can make a good idea see light of day
Five examples

- Optimistic Fair Exchange
- Generic Authentication Architecture
- Channel Binding in Protocol Composition
- Secure Device Pairing
- On-board Credentials
On-board Credentials

Can we safely open up widely deployed secure hardware on mobile devices for use by app developers?
Authentication on the Internet

Username/password rules the Internet
• Cheap, easy-to-deploy, portable
• Annoying, vulnerable (phishing, dictionary attacks, password-stealing trojans…)

Attempts to improve usability and security
• Password-managers
• Single Sign-On
• Better protocols
Hardware tokens

Deployed for specific-services
- More secure, sometimes more intuitive
- More expensive, usually no trusted path to user,
- Single-purpose or issuer-controlled

SW-only credentials

HW credentials
Trusted hardware is widely deployed

- Trusted Execution Environments on smartphones have been available for years
  - Introduced for manufacturer and operator needs
  - Not accessible for app developers

On-board Credentials

An open credential platform that leverages existing mobile TEEs

Secure yet inexpensive
Centralized vs. open provisioning

Centralized provisioning (smart cards)

Open provisioning (On-board Credentials)
On-board Credentials (ObC) architecture

- **Mobile device**
  - Rich execution environment (REE)
    - App
    - ObC API
      - Provisioning, execution, sealing
    - ObC scheduler
      - Trusted app persistent store
      - Trusted app dynamic state
  - Mobile OS

- **Trusted execution environment (TEE)**
  - ObC Interpreter
    - I/O data
    - Interpreted code
    - Interpreter state
    - Loaded trusted app
  - Device key & Device cert

- **Driver**

- **Mobile device hardware with TEE support**
ObC Provisioning (1/2)

Basic Idea: the notion of a family of credential secrets and credential programs endorsed to use them
Open provisioning model

1. Certified device key + user authentication
   PK

2. Provision new family
   Enc(PK, FK)

3. Provision new secrets
   AuthEnc(FK, secret)

4. Provision trusted applications
   AuthEnc(FK, hash(app)) + app


ObC: the aftermath

- Initial prototypes ca. 2008
  - RSA SecurID, SoftSIM
- (Silently) deployed in recent Lumia devices
  - Used for, e.g., MirrorLink attestation
- Stumbling blocks:
  - “who takes liability?” “avoid stepping on toes”
- Related recent standardization
  - Global Platform device committee
  - Open provisioning is elusive

[GP12] “A New Model: The Consumer-Centric Model and How It Applies to the Mobile Ecosystem”
ObC: Lessons Learned

• Address pain points - builds credibility with stakeholders
• Politics can suffocate a good idea
• Standardization can make a good idea see light of day
Lessons Learned

• How to choose the “right” problems?
  – Don’t just guess security requirements; Ask stakeholders
  – Desiderata for deployment and research can be different
  – “90-10 rule” applies to deploying security

• How to identify “good” results?
  – Negative results are useful for security practitioners
  – Capturing researcher interest → (Tech transfer) Impact
  – (Tech transfer) Impact → Capturing researcher interest

• How to find paths to deployment?
  – Address pain points - builds credibility with stakeholders
  – (Standardization) Politics can suffocate a good idea
  – Standardization can make a good idea see light of day