Cryptography & Computer Security

Dieter Gollmann
Hamburg University of Technology
About the Speaker

You were once a cryptographer but now you are a reformed character.
Agenda

- Introducing cryptography
- Know Thyself
- Paradigms for cryptographic computer security services
- Keys that speak for/by themselves
- Analyzing security protocols – theory and practice
History of Ideas

- Crypto had an early start in IT security education.
  - Because it lends itself to academic teaching, pleasingly brief problem descriptions, intellectually challenging solutions?
  - As opposed to computer security; messy problem descriptions, actually building real solutions is tedious.

- One often encounters the view that crypto provides “strong” security compared to other techniques.
History of Ideas – Crypto

- Crypto has its origin in communications security.
- There is a sender and a receiver.
- The communications network is insecure.
- Sender and receiver construct secure logical tunnels.
  - With symmetric crypto, they must share secret keys.
  - With asymmetric cryptography, they need the authentic public key of their peer, e.g. provided by a Public Key Infrastructure.
- Cryptography does not solve security problems; cryptography transforms security problems into key management problems.
Computer Security 101

- **Confidentiality**: crypto has a solution – encryption mechanisms

- **Integrity**: crypto has a solution – message authentication codes, digital signatures
  - These mechanisms can also be used to authenticate a peer.

- **Availability**: crypto is a problem – cryptographic operations need computational and communications resources.
Know Thyself

- In communications security, you authenticate your peer.
- In computer security, you may want to authenticate yourself.
  - “Egress filter”: ensure that a request you are sending out has been created by yourself and not been slipped in by the adversary.
  - “Ingress filter”: ensure that a response you are receiving matches a request you had sent out earlier.
- “Know Thyself” as a new basic security mechanism?
Know Thyself – Cookies

- TCP SYN flooding attack:
  - Attacker sends lots of SYN requests.
  - Server replies with SYN-ACK messages, stores sequence number expected in the final ACK message.
  - Eventually server runs out of resources for dealing with half open connections.

- Solution: do not keep state locally, send the state in the challenge (sequence number).

- Construct cookies from a secret key shared with nobody and relevant session parameters.
Know Thyself – RequestRodeo

- Client-side defence against CSRF attacks.
  - Attacker inserts request in existing authenticated session.

- Proxy between browser and network marks URLs in incoming web pages with unpredictable tokens; for each token stores name of host URL had come from.

- Checks all outgoing requests:
  - URL without a token must have been created locally; can be securely sent in current session.
  - URL with a token sent back to host it is associated with satisfies Same Origin Policy; can be sent in current session.
  - Otherwise, remove all authenticators (cookies) from URL; does not work with SSL sessions.
Paradigms

- Cryptography uses paradigms from the physical world to explain its services.
  - E.g. digital signatures as the equivalent of handwritten signatures for the digital world.
  - Whether this explanation is helpful is another matter.

- Paradigms for crypto services in computer security:
  - Vault
  - Private letter box
  - Transparent vault
Crypto & Computer Security

- **Vault** for locking away sensitive data.
  - Has to be unlocked with a key when putting data in or taking data out.
  - Implemented by symmetric encryption mechanisms.

- **Private letter boxes.**
  - Letter box needs some serial number (public key) so that you can distinguish between letter boxes.
  - Anybody can drop documents into the letter box.
  - Only the owner can open the letter box with a private key.
Crypto & Computer Security

- **Transparent vault**, consider e.g. public lottery draws.
- Everyone can see what is in the vault; only authorized personnel may put items in the vault.
- Private key required for putting items in the vault.
- If the vault has a unique serial number (public key), everyone can refer to items in the vault by this serial number.
- Can create **protected name spaces**; public key is like a database key for organizing and addressing items.
. NET Strong Names

- Assemblies protected by digital signatures:
  - Publisher’s public key given in metadata.
  - Digital signature computed and written into assembly during compilation.
  - Provides origin authentication (w.r.t. name space and data integrity).

- The public key is in fact the ‘identity’ of the publisher.

- Strong names: public key cryptography without a Public Key Infrastructure.

- Method for locally creating globally unique names nobody else can use.
Ownership of Addresses

- **Cryptography Generated Addresses**: proving ownership of dynamically allocated (IPv6) addresses.

- Address owner creates a public key/private key pair; hash of public key is interface ID in IPv6 address.

- Address claim signed with the owner’s private key, signed claim sent together with public key to verifier.

- Verifier checks that the public verification key is linked to the IP address.

- We use public key cryptography without using a PKI.

- Address is the “certificate” for its public key.
Analyzing Security Protocols
Theory & Practice
Cultures in Cryptography

- Theoreticians: ... address theoretical questions as opposed to real world problems ...
  - Try to make protocols secure independent of the implementation.

- Practitioners: ... perspective of specification document writers and that of the implementers ...
  - Try to have secure implementations of protocols.

Protocol design – theory

- Start from abstract specification of the protocol.
- Prove security for abstract specification.
- Ensure that implementation does not introduce vulnerabilities.
- Secure implementation of provably secure protocols.
- Problem: even when the implementation is “secure by design”, the proof of security takes place again in an abstract model; attacks may be possible by exploiting features outside the model.
Example for this approach

- “If you prove something about the (self-identified) cryptographic core of an authentication protocol, does this actually prove *anything* about the full-fledged scheme?”

- “In our model, compactly described in pseudocode, a protocol core (PC) will call out to protocol details (PD), but, for defining security, such calls will be serviced by the adversary.”

[Rogaway, Stegers: Authentication without Elision]
Protocol design – practice

- Case study: protocols for the German eHealth card
- Protocols run between a reader and a card.
  - Card is “passive”; all protocol runs must be initiated by the reader.
- Based on CWA 14980-1 [CEN]:
  - Focus on interoperability, mainly interface specifications.
  - Internal checks in a protocol run not completely specified; this is by intent: do not restrict design space unnecessarily.
- Instruction set from ISO/IEC 7816-4
Case study: ISO 9798-2

- “B verifies TokenAB by deciphering the enciphered part and checking the correctness of the distinguishing identifier B, if present, and that the random number $R_B$, sent to A in step (1), agrees with the random number contained in TokenAB.”

- “Distinguishing identifier B is included in TokenAB to prevent a so-called reflection attack.”
S = RND.IFD||SN.IFD||RND.ICC||SN.ICC||K_{IFD}

R = RND.ICC||SN.ICC||RND.IFD||SN.IFD||K_{ICC}

decrypts input; compares RND.ICC with previous response; verifies RND.ICC, SN.ICC

card reader
interface device

GET DATA

SN.ICC

GET CHALLENGE (n)

RND.ICC

eK_{ENC}(S)||MAC(K_{MAC}:eK_{ENC}(S))

eK_{ENC}(R)||MAC(K_{MAC}:eK_{ENC}(R))

RND ... random number
SN ... serial number

smart card
integrated circuit card
Problem?

Attacker\hspace{1cm} Malou\hspace{1cm} ICC

\textsc{GET DATA}\hspace{1cm} \textsc{SN.ICC}\hspace{1cm} \textsc{GET CHALLENGE (1)}

\begin{align*}
\text{S} &= \text{RND.IFD}||\text{SN.IFD}||\text{RND.ICC}||\text{SN.ICC}||K_{\text{IFD}} \\
\text{eK}_{\text{ENC}}(\text{S}) &|\text{MAC}(K_{\text{IFD}} || \text{RND.IFD} || \text{SN.IFD} || \text{SN.ICC} || K_{\text{IFD}}) \\
\text{eK}_{\text{ENC}}(\text{R}) &|\text{MAC}(K_{\text{IFD}} || \text{RND.IFD} || \text{SN.IFD} || \text{SN.ICC} || K_{\text{IFD}})
\end{align*}

\text{Attacker asks for one byte random challenge; standard does not define how card should react. Don’t trust your inputs!}

\textit{decrypts input; compares RND.IFC with previous response; verifies RND.IFC, SN.ICC}

\textit{R = RND.IFC||SN.IFC||RND.IFD||SN.IFD||K_{IFC}}
Software security

- Software is secure if it can deal with intentionally malformed input.
- In this case, the attacker does not know the secret key and tries to improve her chances of guessing a correct answer by asking for a short challenge.
- Secure software must check its inputs; can then reject or ignore illegal inputs.
- Such a check can be easily implemented on the card but is not prescribed by the standard.
... Variation

R = RND.IFD||SN.IFD||RND.ICC||SN.ICC||K_{IFD}

S = RND.ICC||SN.ICC||RND.IFD||SN.IFD||K_{ICC}
Problem (reflection attack)?

Attacker -> Malou -> ICC:

RND.IFD || SN.ICC

\[ e_{K_{ENC}(S)} || MAC(K_{MAC} \cdot e_{K_{ENC}(S)}) \]

 ICC:

S = RND.ICC ||
SN.ICC ||
RND.IFD ||
SN.ICC || K_{ICC}

ICC does not recognize its own message

Attack succeeds if RND.ICC = RND.IFD

smart card integrated circuit card
On the use of XOR

- XOR with a random value guarantees randomness??
- $K_{\text{ICC}}, K_{\text{IFD}}$ are 32 byte random values.
- $K_{\text{ICC}} \oplus K_{\text{IFD}}$ is input for generation of the session key.
- In the previous scenario $K_{\text{ICC}} = K_{\text{IFD}}$.
- Attacker doesn’t know $K_{\text{ICC}}$, but knows $K_{\text{ICC}} \oplus K_{\text{ICC}} = 0$ and can compute the session key.
- XOR with random value doesn’t give perfect security.
- Use hash function instead and derive session key from $h(K_{\text{ICC}}, K_{\text{IFD}})$. 
Remark

- These are instances of known problems.
- There exist well known and simple fixes.
- Smart cards on the market today may well defend against these attacks.
- How can a decision maker be sure?
- How can a certification body be sure that all relevant undocumented requirements are met by a card?
Conclusion

- Secure implementation of insecure protocols.
- Formal analysis of the protocols discussed previously would flag vulnerabilities.
- Formal analysis needs to be applied to protocol + (partial) card specification; may need to consider specific properties of a cryptographic algorithm.
- Formal analysis needs to consider software security.
- Thank you very much for your attention.