First steps towards cryptographically sound confidentiality analysis of cryptographic protocols

Peeter Laud
peeter.l@ut.ee

Tartu Ülikool
Cybernetica AS
Overview

- Cryptographic protocols.
  - Introduction.
  - Running example.
  - Semantics.
- Security definition.
- Simple analysis.
- Main idea.
- Elaboration on the basis of the running example.
  - Modifying the protocol.
  - (Abstractly) interpreting the protocol.
Cryptographic protocols — structure

- A protocol is a set of roles.
- A role is a sequence of statements.
  - Statements — send and receive messages, construct new messages, take existing messages apart, check the equality of messages.
- Each role also has a name.
  - “Initiator”, “responder”, “server”, etc.
Example protocol

A has a message \( M \), it wants to send it securely to \( B \).

1. \( A \rightarrow S : A, B, N_A \)
2. \( S \rightarrow A : \text{encr}_{K_{AS}}(N_A, B, K_{AB}, \text{encr}_{K_{BS}}(K_{AB}, A)) \)
3. \( A \rightarrow B : \text{encr}_{K_{BS}}(K_{AB}, A) \)
4. \( A \rightarrow B : \text{encr}_{K_{AB}}(M) \)

- \( K_{AS} \) [resp. \( K_{BS} \)] is the shared key between \( A \) [resp. \( B \)] and the server \( S \).
- \( K_{AB} \) is a new key generated by the server.
- \( N_A \) is a nonce — a random number.
More formal write-up

\[
A \\
\text{Generate random } N_A^{(A)} \\
\text{Send } (A, B, N_A^{(A)}) \\
\text{Receive } msg_2 \\
\text{for} A^{(A)} := decr_{K_{AS}}(msg_2) \\
N_A^{(A2)} := \pi_1(forA^{(A)}) \\
\text{Check if } N_A^{(A)} = N_A^{(A2)} \\
K_{AB}^{(A)} := \pi_3(forA^{(A)}) \\
\text{for} B^{(A)} := \pi_4(forA^{(A)}) \\
\text{Send } forB^{(A)} \\
eM := encr_{K_{AB}^{(A)}}(M) \\
\text{Send } eM
\]

\[
B \\
\text{Recieve } msg_3 \\
\text{for} B^{(B)} := decr_{K_{BS}}(msg_3) \\
K_{AB}^{(B)} := \pi_1(forB^{(B)}) \\
\text{Recieve } msg_4 \\
M^{(B)} := decr_{K_{AB}^{(B)}}(msg_4)
\]

\[
S \\
\text{Receive } msg_1 \\
N_A^{(S)} := \pi_3(msg_1) \\
\text{Generate key } K_{AB} \\
\text{for} B^{(S)} := encr_{K_{BS}}(K_{AB}, A) \\
\text{for} A^{(S)} := encr_{K_{AS}}(N_A^{(S)}, B, K_{AB}, forB^{(S)}) \\
\text{Send } forA^{(S)}
\]
Semantics — computation

- All values are bit-strings.
- An encryption scheme — a triple of algorithms \((G, E, D)\) is given.
  - All algorithms here and later are probabilistic polynomial-time (PPT).
- Key generation, encryption and decryption is done by the algorithms \(G, E, D\).
- If “Check if . . .” fails, then the protocol party gets stuck.
- If decryption fails (encryption is not necessarily surjective) or projection fails, then the party gets stuck.
Semantics — communication

All communication is under the control of the adversary — a PPT algorithm.

Sending means handing the message over to the adversary.

Receiving waits, until the adversary provides it with some message.
Security definition

$M$ remains confidential, if

$$(M, \text{view}_{\text{Adv}}(M)) \simeq (M', \text{view}_{\text{Adv}}(M))$$
A very simple-minded analysis

\[
\text{tainted}(M)
\]

\[
x := \text{Expr}(x_1, \ldots, x_k)
\]

\[\exists i : \text{tainted}(x_i) \implies \text{tainted}(x)\]

if \[\exists (\text{Send } y) : \text{tainted}(y)\], then protocol is insecure, otherwise it is secure.

Makes no use of the security properties of encryption...
Security against chosen-ciphertext attack

\((G, E, D)\) is **secure against CCA**, if no PPT algorithm \(A\) can distinguish the following:

- Pair of black boxes \((E_k(\cdot), D_k(\cdot))\), where \(k\) is generated by \(G\) (we denote this \(k \leftarrow G\)).

- Algorithm \(A\) can access these black boxes through **oracle interface** — it can make queries to them.

- Pair of black boxes \((E_k(0), D_k(\cdot))\), where \(k \leftarrow G\).

- 0 is a fixed bit-string.

- When queried, \(E_k(0)\) discards its input.

Under the condition that \(A\) does not query \(D_k(\cdot)\) with anything outputted by the other black box.
Main idea

- We could replace some $enr_K(x)$ with $enr_K(Z)$.
  - $Z$ is such, that $[Z] = 0$.

- This would reduce the dependencies in the analysis.
  - The analysis may give more interesting information about the modified protocol.

- If certain conditions are satisfied then the distributions of $(M, \text{view}_\text{Adv}(M))$ and $(M', \text{view}_\text{Adv}(M))$ do not significantly change.
  - In this case, anything that the analysis claims about the modified protocol is also true for the original protocol.
“Certain conditions”

- Key $k$ must be replacable by $\mathcal{E}_k(\cdot)$ and $\mathcal{D}_k(\cdot)$.

- In construction of messages that are sent out, the key $k$ may only be used as an encryption key.

- May be determined similarly to “tainted”.

- We must know exactly, where the key $k$ is used.

- Key $k$ may occur under several names.

- We’ll elaborate on it later.

- We must make sure that $\mathcal{D}_k(\cdot)$ is not queried with non-allowed values.

- A program transformation helps.
On querying the decryption oracle

Let the uses of $E_k(\cdot)$ [before evaluating $decr_k(w)$] be

\[x_1 := encr_{k_1}(y_1), \ldots, x_n := encr_{k_n}(y_n)\]

Replace $decr_k(w)$ by

\[
\text{case } w \text{ of} \\
\quad x_1 \rightarrow y_1 \\
\quad \ldots \ldots \ldots \\
\quad x_n \rightarrow y_n \\
\quad \text{else } \rightarrow decr_k(w)
\]

For not creating circular dependencies, we consider all serialisations of the protocol.
Example protocol — a serialisation

A: Generate random $N_A^{(A)}$
A: Send $(A, B, N_A^{(A)})$
S: Receive $msg_1$
S: $N_A^{(S)} := \pi_3(msg_1)$
S: Generate key $K_{AB}$
S: $tmp_1 := (K_{AB}, A)$
S: $forB^{(S)} := encr_{K_{BS}}(tmp_1)$
S: $tmp_2 := (N_A^{(S)}, B, K_{AB}, forB^{(S)})$
S: $forA^{(S)} := encr_{K_{AS}}(tmp_2)$
S: Send $forA^{(S)}$
A: Receive $msg_2$
A: $forA^{(A)} := decr_{K_{AS}}(msg_2)$

A: $N_A^{(A2)} := \pi_1(forA^{(A)})$
A: Check if $N_A^{(A)} = N_A^{(A2)}$
A: $K_{AB}^{(A)} := \pi_3(forA^{(A)})$
A: $forB^{(A)} := \pi_4(forA^{(A)})$
A: Send $forB^{(A)}$
B: Receive $msg_3$
B: $forB^{(B)} := decr_{K_{BS}}(msg_3)$
B: $K_{AB}^{(B)} := \pi_1(forB^{(B)})$
A: $eM := encr_{K_{AB}^{(A)}}(M)$
A: Send $eM$
B: Receive $msg_4$
B: $M^{(B)} := decr_{K_{AB}^{(B)}}(msg_4)$
The adversary schedules...  

Is the following case possible?  
1. $M$ remains confidential in all serialisations.  
2. The schedule itself depends on $M$ (and leaks something about it).  

Answer: no.  
1. The schedule depends only on adversary’s actions...  
2. which depend only on adversary’s input...  
3. which is independent of $M$.  

Teooriapäevad Arulas, 3.-5.02.2003 – p.15/27
Example: Using keys

Generate random $N_A^{(A)}$
Send $(A, B, N_A^{(A)})$
Receive $msg_1$
$N_A^{(S)} := \pi_3(msg_1)$
Generate key $K_{AB}$
$\text{tmp}_1 := (K_{AB}, A)$
for $B^{(S)} := \text{encr}_{K_{BS}}(\text{tmp}_1)$
$\text{tmp}_2 := (N_A^{(S)}, B, K_{AB}, B^{(S)})$
for $A^{(S)} := \text{encr}_{K_{AS}}(\text{tmp}_2)$
Send $A^{(S)}$
Receive $msg_2$
for $A^{(A)} := \text{decr}_{K_{AS}}(msg_2)$
$N_A^{(A2)} := \pi_1(\text{for}A^{(A)})$
Check if $N_A^{(A)} = N_A^{(A2)}$
$K_{AB}^{(A)} := \pi_3(\text{for}A^{(A)})$
for $B^{(A)} := \pi_4(\text{for}A^{(A)})$
Send for $B^{(A)}$
Receive $msg_3$
for $B^{(B)} := \text{decr}_{K_{BS}}(msg_3)$
$K_{AB}^{(B)} := \pi_1(\text{for}B^{(B)})$
eM := \text{encr}_{K_{AB}^{(A)}}(M)$
Send eM
Receive $msg_4$
$M^{(B)} := \text{decr}_{K_{AB}^{(B)}}(msg_4)$
Example: Using $K_{AS}$

Generate random $N_A^{(A)}$
Send $(A, B, N_A^{(A)})$
Receive $msg_1$

$N_A^{(S)} := \pi_3(msg_1)$

Generate key $K_{AB}$

tmp$_1 := (K_{AB}, A)$

for$B^{(S)} := \text{encr}_{K_{BS}}(\text{tmp}_1)$

tmp$_2 := (N_A^{(S)}, B, K_{AB}, \text{forB}^{(S)})$

for$A^{(S)} := \text{encr}_{K_{AS}}(\text{tmp}_2)$

Send for$A^{(S)}$
Receive $msg_2$

for$A^{(A)} := \text{decr}_{K_{AS}}(msg_2)$

$N_A^{(A2)} := \pi_1(\text{forA}^{(A)})$

Check if $N_A^{(A)} = N_A^{(A2)}$

$K_{AB}^{(A)} := \pi_3(\text{forA}^{(A)})$

for$B^{(A)} := \pi_4(\text{forA}^{(A)})$

Send for$B^{(A)}$
Recieve $msg_3$

for$B^{(B)} := \text{decr}_{K_{BS}}(msg_3)$

$K_{AB}^{(B)} := \pi_1(\text{forB}^{(B)})$

e$M := \text{encr}_{K_{AS}^{(A)}}(M)$

Send $eM$
Recieve $msg_4$

$M^{(B)} := \text{decr}_{K_{AB}^{(B)}}(msg_4)$

$K_{BS}$ is not $K_{AS}$.

$K_{AB}^{(?)}$ comes from a message from the network.
Example: replacing $K_{AS}$

Generate random $N_A^{(A)}$
Send $(A, B, N_A^{(A)})$

Receive $msg_1$

$N_A^{(S)} := \pi_3(msg_1)$

Generate key $K_{AB}$

$tmp_1 := (K_{AB}, A)$

$forB^{(S)} := \text{encr}_{K_{BS}}(tmp_1)$

$tmp_2 := (N_A^{(S)}, B, K_{AB}, forB^{(S)})$

$forA^{(S)} := \text{encr}_{K_{AS}}(Z)$

Send $forA^{(S)}$

Receive $msg_2$

$forA^{(A)} := \text{case } msg_2 \text{ of}$

$\text{forA}^{(S)} \rightarrow tmp_2$
$\text{else} \rightarrow \text{decr}_{K_{AS}}(msg_2)$

$N_A^{(A2)} := \pi_1(forA^{(A)})$

Check if $N_A^{(A)} = N_A^{(A2)}$

$K_{AB}^{(A)} := \pi_3(forA^{(A)})$

$forB^{(A)} := \pi_4(forA^{(A)})$

Send $forB^{(A)}$

Recieve $msg_3$

$forB^{(B)} := \text{decr}_{K_{BS}}(msg_3)$

$K_{AB}^{(B)} := \pi_1(forB^{(B)})$

$eM := \text{encr}_{K_{AB}^{(A)}}(M)$

Send $eM$

Recieve $msg_4$

$M^{(B)} := \text{decr}_{K_{AB}^{(B)}}(msg_4)$
Example: replacing $K_{BS}$

Generate random $N_A^{(A)}$
Send $(A, B, N_A^{(A)})$
Receive $msg_1$
$N_A^{(S)} := \pi_3(msg_1)$
Generate key $K_{AB}$
tmp$_1 := (K_{AB}, A)$
forB$^{(S)} := encr_{K_{BS}}(Z)$
tmp$_2 := (N_A^{(S)}, B, K_{AB}, \text{forB}^{(S)})$
forA$^{(S)} := encr_{K_{AS}}(Z)$
Send forA$^{(S)}$
Receive $msg_2$
forA$^{(A)} := \text{case } msg_2 \text{ of}$
  forA$^{(S)} \rightarrow \text{tmp}_2$
  else $\rightarrow \text{decr}_{K_{AS}}(msg_2)$

$N_A^{(A2)} := \pi_1(\text{forA}^{(A)})$
Check if $N_A^{(A)} = N_A^{(A2)}$
$K_{AB}^{(A)} := \pi_3(\text{forA}^{(A)})$
forB$^{(A)} := \pi_4(\text{forA}^{(A)})$
Send forB$^{(A)}$
Receive $msg_3$
forB$^{(B)} := \text{case } msg_3 \text{ of}$
  forB$^{(S)} \rightarrow \text{tmp}_1$
  else $\rightarrow \text{decr}_{K_{BS}}(msg_3)$
$K_{AB}^{(B)} := \pi_1(\text{forB}^{(B)})$
eM := encr_{K_{AB}^{(A)}}(M)$
Send eM
Receive $msg_4$
M$^{(B)} := \text{decr}_{K_{AB}^{(B)}}(msg_4)$
What about $K_{AB}$?

Generate random $N_A^{(A)}$

Send $(A, B, N_A^{(A)})$

Receive $msg_1$

$N_A^{(S)} := \pi_3(msg_1)$

Generate key $K_{AB}$

tmp$_1 := (K_{AB}, A)$

for$B^{(S)} := encr_{K_{BS}}(Z)$

tmp$_2 := (N_A^{(S)}, B, K_{AB}, forB^{(S)})$

for$A^{(S)} := encr_{K_{AS}}(Z)$

Send for$A^{(S)}$

Receive $msg_2$

for$A^{(A)} := case msg_3 of$

for$A^{(S)} \rightarrow tmp_2$

else $\rightarrow decr_{K_{BS}}(msg_3)$

for$B^{(S)} := \pi_3(forA^{(A)})$

for$B^{(A)} := \pi_4(forA^{(A)})$

Send for$B^{(A)}$

Receive $msg_3$

for$B^{(B)} := case msg_2 of$

for$B^{(S)} \rightarrow tmp_1$

else $\rightarrow decr_{K_{BS}}(msg_3)$

for$B^{(A)} := \pi_1(forB^{(B)})$

e$M := encr_{K_{AB}^{(A)}}(M)$

Send e$M$

Receive $msg_4$

$M^{(B)} := decr_{K_{AB}^{(B)}}(msg_4)$
What about $K_{AB}$?

- The variable $K_{AB}$ is not sent out.
- Are $K_{AB}^{(A)}$ and $K_{AB}^{(B)}$ equal to $K_{AB}$?

We “interpret” the protocol, assigning to each variable an abstract value from a term algebra with

- Constant symbols: keys, random values, adversary’s inputs.
- Operators: pairing, projections, encryption, decryption, case-construction.
- Certain cancellation rules.

Cancellation rules and certain assumptions about the inequality of terms allow us to check, whether the keys are equal or not.

All cancellation rules and inequality assumptions are semantically sound.
Interpreting statements

- Statement: \( x := \text{Expr}(x_1, \ldots, x_n). \)
  - The abstract value \( A(x) = \text{Expr}(A(x_1), \ldots, A(x_n)). \)

- Statement: Check if \( x = y. \)
  - First check, whether \( A(x) = A(y) \) is possible.
  - If yes, then replace more complex abstract value with the simpler one.
    - Keys, random values are the simplest.
    - Terms containing adversary’s inputs are the most complex.
  - Do the same replacement (replace one subterm with another) also in the abstract values of other variables.

Am I inventing the bicycle here?
Interpreting *case*-expressions

The statement

\[
z := \text{case } w \text{ of } \\
\quad x_1 \rightarrow y_1 \\
\quad \ldots \ldots \\
\quad x_n \rightarrow y_n \\
\quad \text{else } \rightarrow \text{decr}_K(w)
\]

is replaced with

Check if \( w = x_i \)

\[
z := y_i
\]

\( n \) variants, similar to serialisation.
\[ \text{else } \rightarrow \text{decr}_K(w) \]

An encryption system \( (G, E, D) \) has **ciphertext integrity**, if:

No PPT algorithm \( \mathcal{A} \) with access to oracles \( E_k(\cdot) \) and \( D_k(\cdot) \) can submit to \( D_k(\cdot) \) a bit-string \( y \), such that

- \( D_k(y) \) exists, i.e. \( y \) is a valid ciphertext;
- \( y \) was not an output of \( E_k(\cdot) \).

i.e. there is no \textit{else}-clause.
What about $K_{AB}$?

Generate random $N^{(A)}_A$
Send $(A, B, N^{(A)}_A)$
Receive $msg_1$
$N^{(S)}_A := \pi_3(msg_1)$
Generate key $K_{AB}$
$tmp_1 := (K_{AB}, A)$
forall $B := encr_{K_{BS}}(Z)$
$tmp_2 := (N^{(S)}_A, B, K_{AB}, forall B)$
forall $A := encr_{K_{AS}}(Z)$
Send $forall A$
Receive $msg_2$
Check if $msg_2 = forall A$
forall $A := tmp_2$

Now obviously $K_{AB} = K^{(A)}_{AB} = K^{(B)}_{AB}$. 
$M$ remains confidential

Generate random $N_A^{(A)}$
Send $(A, B, N_A^{(A)})$

Receive $msg_1$
$N_A^{(S)} := \pi_3(msg_1)$
Generate key $K_{AB}$
$\text{tmp}_1 := (K_{AB}, A)$
$\text{forB}^{(S)} := \text{encr}_{K_{BS}}(Z)$

$\text{tmp}_2 := (N_A^{(S)}, B, K_{AB}, \text{forB}^{(S)})$
$\text{forA}^{(S)} := \text{encr}_{K_{AS}}(Z)$
Send $\text{forA}^{(S)}$

Receive $msg_2$
Check if $msg_2 = \text{forA}^{(S)}$
$\text{forA}^{(A)} := \text{tmp}_2$

$N_A^{(A2)} := \pi_1(\text{forA}^{(A)})$
Check if $N_A^{(A)} = N_A^{(A2)}$
$K_{AB}^{(A)} := \pi_3(\text{forA}^{(A)})$
$\text{forB}^{(A)} := \pi_4(\text{forA}^{(A)})$
Send $\text{forB}^{(A)}$

Recieve $msg_3$
Check if $msg_3 = \text{forB}^{(S)}$
$\text{forB}^{(B)} := \text{tmp}_1$
$K_{AB}^{(B)} := \pi_1(\text{forB}^{(B)})$
$eM := \text{encr}_{K_{AB}^{(A)}}(Z)$
Send $eM$

Recieve $msg_4$
$M^{(B)} := \text{case msg}_4 \text{ of}$
eeM $\rightarrow M$

Simple-minded analysis works now.
Conclusions and open questions

- The approach seems to work.
- Can the number of variants needing analysis (through serialisation and interpretation of `case`-expressions) be bounded? Is considering several variants necessary at all?
- How well does finding out the equality of keys work? Are there other approaches? Is it necessary at all?