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.
  - “Initiatior”, “responder”, “server”, etc.

Example protocol

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

1. $A \longrightarrow S : A, B, N_A$
2. $S \longrightarrow A : encr_{K_{AS}}(N_A, B, K_{AB}, encr_{K_{BS}}(K_{AB}, A))$
3. $A \longrightarrow B : encr_{K_{BS}}(K_{AB}, A)$
4. $A \longrightarrow B : encr_{K_{AB}}(M)$

- 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

Generate random $N_A^{(A)}$

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

Receive msg_2

$forA^{(A)} := decr_{K_{AS}}(msg_2)$

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

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

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

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

Send $forB^{(A)}$

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

Send eM

B

Recieve msg_3

$forB^{(B)} := decr_{K_{BS}}(msg_3)$

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

Recieve msg_4

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

S

Receive msg_1

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

Generate key K_{AB}

$forB^{(S)} := encr_{K_{BS}}(K_{AB}, A)$.

$forA^{(S)} := encr_{K_{AS}}(N_A^{(S)}, B, K_{AB}, forB^{(S)})$

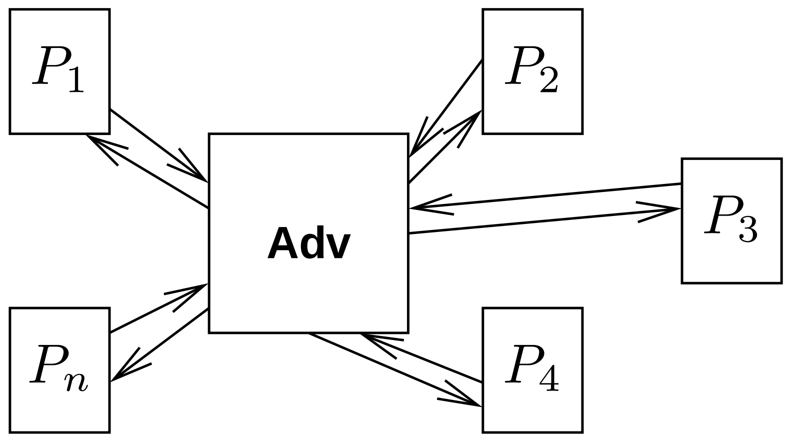
Send $forA^{(S)}$

Semantics — computation

- All values are bit-strings.
- An encryption scheme — a triple of algorithms $(\mathcal{G}, \mathcal{E}, \mathcal{D})$ is given.
  - All algorithms here and later are probabilistic polynomial-time (PPT).
- Key generation, encryption and decryption is done by the algorithms $\mathcal{G}$, $\mathcal{E}$, $\mathcal{D}$.
- 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.

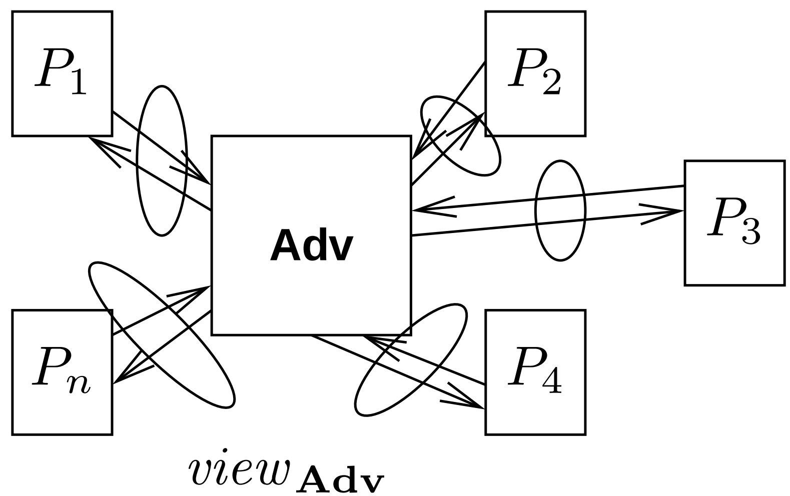


- Sending means handing the message over to the adversary.
- Receiving waits, until the adversary provides it with some message.

Security definition

M remains confidential, if

$$(M, view_{\mathbf{Adv}}(M)) \approx (M', view_{\mathbf{Adv}}(M)) \quad .$$



A very simple-minded analysis

tainted(M)

$$x := \mathrm{Expr}(x_1, \ldots, x_k)$$

$\exists i :$ *tainted*$(x_i) \Longrightarrow$ *tainted*(x)

if $\exists$(Send y) $:$ *tainted*(y), then protocol is insecure, otherwise it is secure.

Makes no use of the security properties of encryption. . .

Security against chosen-ciphertext attack

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

- Pair of black boxes $(\boxed{\mathcal{E}_k(\cdot)}, \boxed{\mathcal{D}_k(\cdot)})$, where k is generated by $\mathcal{G}$ (we denote this $k \leftarrow \mathcal{G}$).
  - Algorithm $\mathcal{A}$ can access these black boxes through oracle interface — it can make queries to them.
- Pair of black boxes $(\boxed{\mathcal{E}_k(\mathbf{0})}, \boxed{\mathcal{D}_k(\cdot)})$, where $k \leftarrow \mathcal{G}$.
  - $\mathbf{0}$ is a fixed bit-string.
  - When queried, $\boxed{\mathcal{E}_k(\mathbf{0})}$ discards its input.

Under the condition that $\mathcal{A}$ does not query $\boxed{\mathcal{D}_k(\cdot)}$ with anything outputted by the other black box.

Main idea

- We could replace some $encr_K(x)$ with $encr_K(Z)$.
  - Z is such, that $[\![Z]\!] = \mathbf{0}$.
- 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 $(M, view_{\mathbf{Adv}}(M))$ and $(M', view_{\mathbf{Adv}}(M))$ 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 $\boxed{\mathcal{E}_k(\cdot)}$ and $\boxed{\mathcal{D}_k(\cdot)}$.
  - 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 $\boxed{\mathcal{D}_k(\cdot)}$ is not queried with non-allowed values.
  - A program transformation helps.

On querying the decryption oracle

Let the uses of $\mathcal{E}_k(\cdot)$ [before evaluating $decr_k(w)$] be

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

Replace $decr_k(w)$ by

$$\begin{array}{l} case\ w\ of \\ \quad x_1 \quad \to y_1 \\ \quad \ldots\ldots\ldots\ldots\ldots \\ \quad x_n \quad \to y_n \\ \quad else \to decr_k(w) \end{array}$$

No change to adversary's view

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: Recieve 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: Recieve msg_4

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

The adversary schedules...

- Is the following case possible?
  - M remains confidential in all serialisations.
  - The schedule itself depends on M (and leaks something about it).
- Answer: no.
  - The schedule depends only on adversary's actions...
  - which depend only on adversary's input...
  - which is independent of M.

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}

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

$forB^{(S)} := encr_{K_{BS}}(tmp_1)$

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

$forA^{(S)} := encr_{K_{AS}}(tmp_2)$

Send $forA^{(S)}$

Receive msg_2

$forA^{(A)} := decr_{K_{AS}}(msg_2)$

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

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

$forB^{(B)} := decr_{K_{BS}}(msg_3)$

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

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

Send eM

Recieve msg_4

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

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}

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

$forB^{(S)} := encr_{K_{BS}}(tmp_1)$

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

$forA^{(S)} := encr_{K_{AS}}(tmp_2)$

Send $forA^{(S)}$

Receive msg_2

$forA^{(A)} := decr_{K_{AS}}(msg_2)$

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

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

$forB^{(B)} := decr_{K_{BS}}(msg_3)$

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

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

Send eM

Recieve msg_4

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

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}

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

$forB^{(S)} := encr_{K_{BS}}(tmp_1)$

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

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

Send $forA^{(S)}$

Receive msg_2

$$forA^{(A)} := case\ msg_2\ of \quad \begin{array}{ll} forA^{(S)} & \rightarrow tmp_2 \\ else & \rightarrow decr_{K_{AS}}(msg_2) \end{array}$$

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

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

$forB^{(B)} := decr_{K_{BS}}(msg_3)$

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

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

Send eM

Recieve msg_4

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

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}

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

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

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

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

Send $forA^{(S)}$

Receive msg_2

$$forA^{(A)} := case\ msg_2\ of \quad forA^{(S)} \to tmp_2 \quad else \to decr_{K_{AS}}(msg_2)$$

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

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

$$forB^{(B)} := case\ msg_3\ of \quad forB^{(S)} \to tmp_1 \quad else \to decr_{K_{BS}}(msg_3)$$

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

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

Send eM

Recieve msg_4

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

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} $\longleftarrow$

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

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

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

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

Send $forA^{(S)}$

Receive msg_2

$$forA^{(A)} := \mathit{case}\ msg_3\ \mathit{of}\quad forA^{(S)} \rightarrow tmp_2;\quad \mathit{else} \rightarrow decr_{K_{AS}}(msg_2)$$

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

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

$$forB^{(B)} := \mathit{case}\ msg_2\ \mathit{of}\quad forB^{(S)} \rightarrow tmp_1;\quad \mathit{else} \rightarrow decr_{K_{BS}}(msg_3)$$

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

$eM := encr_{K_{AB}^{(A)}}(M)$ $\longleftarrow$

Send eM

Recieve msg_4

$M^{(B)} := decr_{K_{AB}^{(B)}}(msg_4)$ $\longleftarrow$

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

$$z := case\ w\ of$$

$$x_1 \;\; \rightarrow y_1$$

$$\dots\dots\dots\dots\dots\dots\dots\dots$$

$$x_n \;\; \rightarrow y_n$$

$$else \rightarrow decr_K(w)$$

is replaced with

Check if $w = x_i$

$$z := y_i$$

n variants, similar to serialisation.

$else \rightarrow decr_K(w)$

An encryption system $(\mathcal{G}, \mathcal{E}, \mathcal{D})$ has ciphertext integrity, if:

No PPT algorithm $\mathcal{A}$ with access to oracles $\boxed{\mathcal{E}_k(\cdot)}$ and $\boxed{\mathcal{D}_k(\cdot)}$ can submit to $\boxed{\mathcal{D}_k(\cdot)}$ a bit-string y, such that

- $\mathcal{D}_k(y)$ exists, i.e. y is a valid ciphertext;
- y was not an output of $\boxed{\mathcal{E}_k(\cdot)}$.

i.e. there is no $else$-clause.

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}

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

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

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

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

Send $forA^{(S)}$

Receive msg_2

Check if $msg_2 = forA^{(S)}$

$forA^{(A)} := tmp_2$

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

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

Check if $msg_3 = forB^{(S)}$

$forB^{(B)} := tmp_1$

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

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

Send eM

Recieve msg_4

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

Now obviously $K_{AB} = K_{AB}^{(A)} = K_{AB}^{(B)}$.

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}

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

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

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

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

Send $forA^{(S)}$

Receive msg_2

Check if $msg_2 = forA^{(S)}$

$forA^{(A)} := tmp_2$

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

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

Check if $msg_3 = forB^{(S)}$

$forB^{(B)} := tmp_1$

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

$eM := encr_{K_{AB}^{(A)}}(Z)$

Send eM

Recieve msg_4

$M^{(B)} := \mathit{case}\ msg_4\ \mathit{of}$

$\quad eM \to M$

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?