# 02244 Language Based Security
# Project on Security Protocols

Hand-out: March 12, 2018
Hand-in: via **Campusnet** until May 7, 2018 **noon**
We allow to work and hand-in in **groups of up to three students**.
Page Limit: 15 pages (not counting any AnB files you attach)

**Presentation:** in the last meeting (on May 7th), we ask every group to present the protocol they have designed, so we can discuss the different solutions, common mistakes, and insights gained – and thus every group gets feedback on their report.

## EasyID

Much of this assignment is a project to design and verify a small security protocol "EasyID" similar to the Danish "NemID", where you are welcome to explore different design choices. The purpose of EasyID is to allow users to authenticate themselves to various websites and make legally binding transactions (like e-Banking) without having to first setup a relation (like a password) between user and website. Such a protocol requires, however, a so-called "trusted third party" to establish a session between user and website; in the case of NemID this is the Nets company that is running NemID. Users must be registered at this trusted third party, having a user name (can be the CPR number), a password, as well as a code card. Moreover, this works only with websites that *trust* the trusted third party, i.e., that accept login via the trusted third party (e.g., outside Denmark there are not many websites that accept NemID).

The typical workflow of NemID is as follows:

- A user establishes a "secure" connection to a website using TLS that offers NemID login with a login-frame of NemID where user name (CPR) and password can be entered.
- This information is forwarded to the Nets-Server; if the password is correct, it replies with a code on the users code card that is presented to the user.
- The user has to look up the code on the code card and reply with the respective entry. We will discuss in class how we can model/represent such a code card in AnB.
- This is again forwarded to NemID and, if correct, NemID informs the server that this indeed the claimed user.

The NemID protocol has often been criticized because the entire communication between user and NemID goes through the website which is problematic if the website is not honest. (What attacks could a dishonest website launch?) You should design your protocol EasyID so that it is secure even if used with a dishonest website.

The **initial knowledge** you may assume about the parties of the protocol:

- The names of all involved participants may be known initially to each other. This is despite the fact that user names may be CPR numbers that are usually considered sensitive; one should thus not transmit them in any way that could be read by anybody except the involved website and the trusted third party.
- You may assume that both the trusted third party and all websites have a public/private key pair, and that all public keys are known to all participants. The user does **not** have a public/private key pair though.
- The user and the trusted third party have two shared secrets the user's password and code card. Depending on how you model the code card, there may be additional items that user and trusted third party need to share.

**Important:** If your intended protocol design requires any additional knowledge beyond the previous list, please do first discuss this with the teacher.

We desire that the protocol should be secure, even though some users or websites may be dishonest. We want to assume only that the trusted third party is necessarily honest. Moreover, we assume the worst case that all dishonest parties work together as "the intruder" who also controls the entire communication medium.

## 1 Syntax & Semantics (Course Week 7-9)

**Task 1** This first task is the design of the protocol. In the second task, you shall formalize it in the AnB language, but it is helpful for this task *not* to think about AnB yet, so you do not get constrained by what you can (easily) express in AnB. So try to become clear here about your protocol and about the fundamental, conceptual questions:

- What will be the "result" of the protocol, i.e., what is an appropriate means for a "secure connection" between a user an a website?
- How can we model a "secure" connections between user, website and trusted third party without modeling the entire TLS protocol? What properties of the connection are really needed for the EasyID protocol? What cryptographic building blocks do we need?

  A good way is to think for any connection why it would need to be encrypted.
- How can we formalize the informal goals for the protocol, in particular as authentication and secrecy problems?
- Does the standard Dolev-Yao model capture the assumptions about the intruder and dishonest participants—if there are differences, what do they mean?

The answer to these questions can be helpful in the following steps, you should review it after task 2, because the experience with OFMC may reveal attacks or insights that you did not see during your first design. You are more than welcome to document also the "evolution" of the protocol in your report and why you made some changes.

**Task 2** The next step is to formalize your protocol in the language AnB and use the OFMC tool to verify it. Since classic OFMC can only verify for a bounded number of sessions, we require here verification for only 2 sessions (i.e., as soon as OFMC outputs `"Verified for 2 sessions."`, you can stop).

Please do make tests/experiments: if you remove one of the features of your protocols, you probably should expect an attack. Check that OFMC indeed finds this attack if you make a change to the protocol. If not, check why an expected attack is not found: there may be a bug in your specification or in OFMC (please report OFMC bugs directly!). Also it may be that some security measure was indeed redundant. For the following tasks it is easier if the protocol is as simple as possible.

In the report, it is crucial that you do *not* simply give an AnB specification as an answer to this task, but that you explain what choices you have made and why—this concerns both choices in the protocol design and choices of modeling. You are most welcome to also describe failed attempts for which OFMC reported an attack, and what that attack means (e.g., an overlooked problem or a nonsensical model).

Please also explicitly give in the report the initial knowledge that the intruder gets according to your protocol specification when there are two honest agents $a$ and $b$ (besides the honest identity provider and the dishonest intruder $i$).

**Task 3** Consider now that many people use *weak passwords*. For a *guessing* attack, the intruder compiles a list of strings that are popular passwords (like first names, 123 etc), and also automatically creates a number of variants (flipping lower case/upper case, adding a digit). This list is called a *dictionary* and it is typically in the order of thousands or millions of entries, so it is feasible automatically perform a few steps for each entry of the dictionary.

Is your protocol "safe" against dictionary attacks? In what sense could you argue for what "safe" means here? Note that you cannot directly verify this in OFMC.

**Task 4** An intruder knows the following messages ($h \in \Sigma_p$ is a public function):

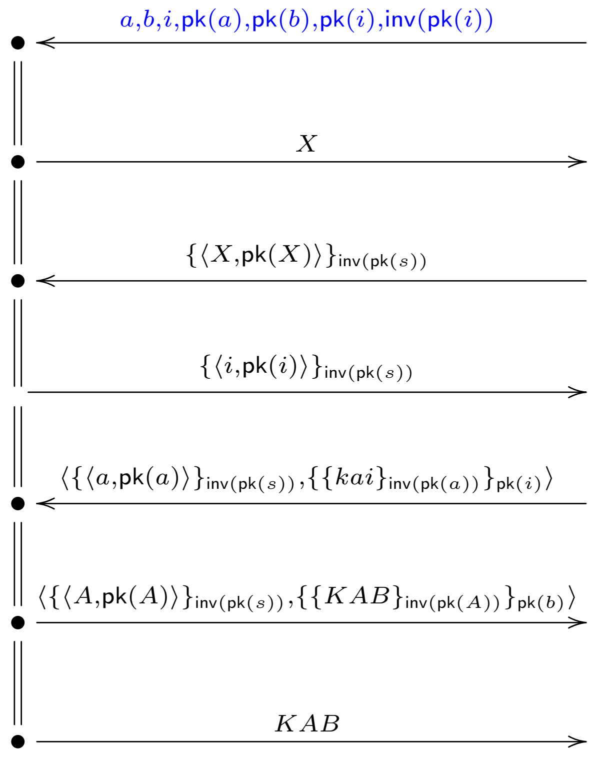
$$g, pk(a), pk(i), inv(pk(i)),$$
$$\{x_1\}_{pk(i)}, \{\!|k_1|\!\}_{k_2}, \{\!|k_2|\!\}_{\exp(\exp(g,x_1),x_2)}, \{\!|n_1|\!\}_{h(k_1,k_2)}, \{\exp(g,x_2)\}_{\mathsf{inv}(pk(a))}$$

Which atomic messages can the intruder derive? Give proofs (derivation trees) for each atomic message that the intruder can derive using the intruder deduction rules of the lecture.

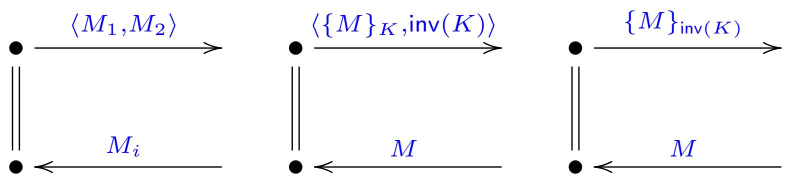
**Task 5** Translate your protocol from task 2 into strands to represent the different roles.

## 2 The Lazy Intruder (Course Week 10)

**Task 6** Consider the intruder constraint:

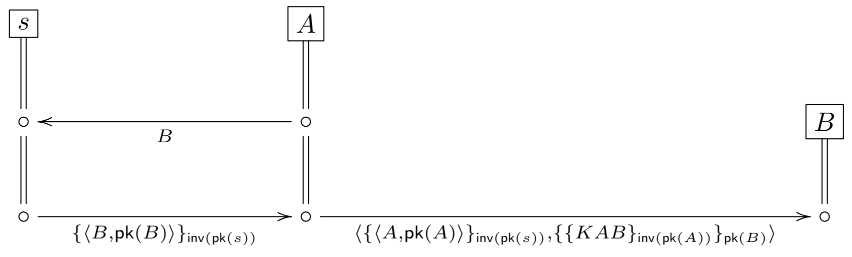

$a, b, i, \mathsf{pk}(a), \mathsf{pk}(b), \mathsf{pk}(i), \mathsf{inv}(\mathsf{pk}(i))$
$X$
$\{\langle X, \mathsf{pk}(X)\rangle\}_{\mathsf{inv}(\mathsf{pk}(s))}$
$\{\langle i, \mathsf{pk}(i)\rangle\}_{\mathsf{inv}(\mathsf{pk}(s))}$
$\langle \{\langle a, \mathsf{pk}(a)\rangle\}_{\mathsf{inv}(\mathsf{pk}(s))}, \{\{kai\}_{\mathsf{inv}(\mathsf{pk}(a))}\}_{\mathsf{pk}(i)}\rangle$
$\langle \{\langle A, \mathsf{pk}(A)\rangle\}_{\mathsf{inv}(\mathsf{pk}(s))}, \{\{KAB\}_{\mathsf{inv}(\mathsf{pk}(A))}\}_{\mathsf{pk}(b)}\rangle$
$KAB$

Recall that the intruder can at any point perform analysis steps according to the following rules (where $i \in \{1, 2\}$):


$\langle M_1, M_2\rangle$
$M_i$
$\langle \{M\}_K, \mathsf{inv}(K)\rangle$
$M$
$\{M\}_{\mathsf{inv}(K)}$
$M$

1. Show that this constraint is satisfiable, using the lazy intruder reduction rules.
2. Check: is there more than one solution? For this, check all steps of your previous reduction and check if any alternative derivation is possible and whether that leads to a solution.
3. The constraint arises from the following protocol (except for the first step in blue,

which gives the initial intruder knowledge):


$s$
$A$
$B$
$B$
$\{\langle B,\mathsf{pk}(B)\rangle\}_{\mathsf{inv}(\mathsf{pk}(s))}$
$\langle\{\langle A,\mathsf{pk}(A)\rangle\}_{\mathsf{inv}(\mathsf{pk}(s))},\{\{KAB\}_{\mathsf{inv}(\mathsf{pk}(A))}\}_{\mathsf{pk}(B)}\rangle$

How can this protocol lead to the constraint above, i.e., why is $M_0$ a possible initial knowledge and relate each line in the constraint to a particular protocol step and explain what is happening.

4. Consider the goal $KAB$ secret between $A, B$. Does a solution to the constraint represent an attack against this goal?

## 3 Abstraction (Course Week 11)

**Task 7** In the lecture, we discuss how to encode a protocol into intruder-deduction rules where freshly created data is abstracted. Apply this technique to your original protocol from Task 2.

- What could be a suitable abstraction of freshly generated keys (and nonces if there are)?
- Describe a set of intruder Horn clauses for your protocol. Check these clauses with your teacher before the next step.
- Do these deduction rules allow the intruder to deduce any secret key?

## 4 Channels & Composition (Course Week 12)

**Task 8** Reformulate the protocols from tasks 1 and 2 using channels (as assumptions) replacing cryptography as far as possible with channels. What are the "weakest" forms of channels that you need to assume in which case? Is it possible to use TLS for establishing the channels you have assumed?