Description of some case studies

Stéphanie Delaune and Lucca Hirschi

LSV, ENS Cachan & CNRS & INRIA Saclay Île-de-France

Abstract. Privacy is a general requirement that needs to be studied in different contexts. In a previous report, we identify some applications for wich privacy plays an important role, and with significant interest in terms of societal impact.

In this report, we describe some case studies that we will use as a guideline for our research agenda. Our goal is to establish a repository of protocols that are representative of the selected applications chosen in Task 2. We decide to concentrate our efforts on electronic voting protocols and RFID protocols. In addition, we consider two real case studies: the ICAO standard that specifies the protocols involved in the e-passport application, and the UMTS standard used in 3G mobile phone systems.

1 Some electronic voting protocols

Privacy-type properties play an important role in e-voting protocols. We consider two protocols that rely on different mechanisms to ensure some privacy-type properties such as anonymity or receipt-freeness. They all involve some unusual cryptographic primitives and can not be analyse using APTE [5] or SPEC [11]. Moreover, the equivalence checked by Proverif is too strong. The only existing tool that is able to handle these examples (or at least a simplified version of these protocols) is AKISS [4].

The two protocols we consider highly rely on two specific cryptographic schemes: bit-commitment and blind signature. For modelling these schemes, we must deal with complex algebraic relations which are out of the scope of existing tools except AKISS.

Bit-commitment. This scheme allows a voter to commit a message containing an hidden value to an agent such that (i) the voter can reveal the value later on; (ii) the agent is ensured that the revealed value is the same as the one contained in the message.

Blind Signature. With this scheme, an authority S is able to blindly sign a message created by a voter. S sends a blind signature to the voter who is then able to use it to sign a message. The agent S can now verify the signature but the message remains hidden.

1.1 FOO protocol

In this section we give an informal description of a protocol due to Fujioka, Okamoto and Ohta [7]. The protocol involves voters, an administrator, verifying that only eligible voters can cast votes, and a collector, collecting and publishing the votes. The whole protocol is summarized in Figure 1.

Phase 1. In a first phase, the voter gets a signature on a commitment to his vote from the administrator. To ensure privacy, blind signatures are used, i.e. the administrator does not learn the commitment of the vote.

- Voter V selects a vote v and computes the commitment $x = \mathsf{commit}(v, r)$ using a random key r;
- V computes the message $e = \mathsf{blind}(x, b)$ using a random blinding factor b;
- V digitally signs e and sends her signature $\mathsf{sign}(e, \mathsf{priv}(V))$ to the administrator A together with her identity;
- A verifies that V has the right to vote, has not voted yet and that the signature is valid; if all these tests hold, A digitally signs e and sends his signature $\mathsf{sign}(e, \mathsf{priv}(A))$ to V;

Phase 2. The second phase of the protocol is the actual voting phase.

- V now *unblinds* $\mathsf{sign}(e, \mathsf{priv}(A))$ and obtains $y = \mathsf{sign}(x, \mathsf{priv}(A))$, i.e. a signed commitment to V's vote.
- V sends y, A's signature on the commitment to V's vote, to the collector C using an anonymous channel;
- C checks correctness of the signature y and, if the test succeeds, enters (ℓ, x, y) into a list as an ℓ-th item.

Phase 3. The last phase of the voting protocol starts, once the collector decides that he received all votes, e.g. after a fixed deadline. In this phase the voters reveal the random key r which allows C to open the votes and publish them.

- C publishes the list (ℓ_i, x_i, y_i) of commitments he obtained;
- V verifies that her commitment is in the list and sends ℓ, r to C via an anonymous channel;
- C opens the ℓ-th ballot using the random r and publishes the vote v.

Vote privacy. This scheme has been shown to satisfy the notion of privacy. To ensure privacy, secrecy of the keys are not needed, and actually it is not necessary to make some assumptions about the correctness of the administrator or the collector, who may be corrupt. It is however important to ensure that voters use the same public key for the administrator.

The use of phases is also crucial for privacy to be respected. When we omit the synchronisation after the registration phase with the administrator, privacy is violated. Indeed, consider the following scenario with two voters V_A and V_B. Voter V_A contacts the administrator. As no synchronisation is considered, voter V_A can send his committed vote to the collector before voter V_B contacts the administrator. As voter V_B could not have submitted the committed vote, the attacker can link this commitment to the first voters identity.

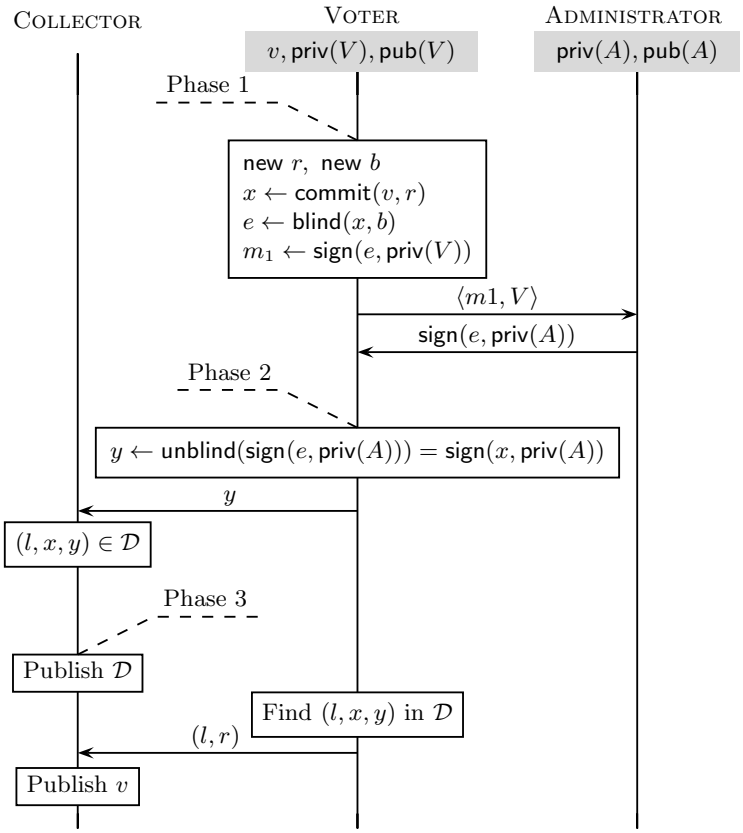

Fig. 1: FOO protocol

Receipt-freeness. This scheme is not receipt-free and was in fact not designed with receipt-freeness in mind. Indeed, if the voter gives away the random numbers for blinding and commitment, the coercer can verify that the committed vote corresponds to the coercers wish and by unblinding the first message, the coercer can trace which vote corresponds to this particular voter. Moreover, the voter cannot lie about these values as this will immediately be detected by the coercer.

1.2 Okamoto protocol

In this section we study a protocol due to Okamoto [9] which was designed to be incoercible. However, Okamoto himself shows a flaw [10].

The authorities managing the election are an administrator for registration, a collector for collecting the tokens and a timeliness member (denoted by T) for

publishing the final tally. The main difference with the protocol due to Fujioka *et al.* is the use of a trap-door bit commitment scheme in order to retrieve receipt-freeness. Such a commitment scheme allows the agent who has performed the commitment to open it in many ways. Hence, trap-door bit commitment does not bind the voter to the vote v. Now, to be sure that the voter does not change her mind at the end (during the opening stage) she has to say how she wants to open her commitment during the voting stage. This is done by sending the required information to T through an untappable anonymous channel, i.e. a physical apparatus by which only voter V can send a message to a party, and the message is perfectly secret to all other parties. The figure 2 describes the protocol.

Phase 1. The first phase is similar to the one of the protocol due to Fujioka *et al.*. The only change is that $\mathsf{tdcommit}$ is a trap-door bit commitment scheme.

Phase 2. The second phase of the protocol is the actual voting phase. Now, the voter has to say how she wants to open her commitment to the timeliness member T.

- V sends y, A's signature on the trap-door commitment to V's vote, to the collector C using an anonymous channel;
- C checks correctness of the signature y and, if the test succeeds, enters (x, y) into a list.
- V sends (v, r, x) to the timeliness member T through an untappable anonymous channel.

Phase 3. The last phase of the voting protocol starts, once the collector decides that he received all votes, e.g. after a fixed deadline.

- C publishes the list (x_i, y_i) of trap-door commitments he obtained;
- V verifies that her commitment is in the list;
- T publishes the list of the vote v_i in random order and also proves that he knows the permutation π and the r_i's such that $x_{\pi(i)} = \mathsf{tdcommit}(v_i, r_i)$ without revealing π or the r_i's.

Vote privacy. Privacy can be established as in the previous protocol.

Receipt-freeness. This protocol has been shown to be receipt-free. The idea is that it is now possible for a voter to vote a (the vote of his choice), but when outputting secrets to the coercer, he lies and gives him fake secrets to pretend to cast the vote c (the choice of the coercer). The crucial part is that, using trap-door commitment and thanks to the fact that the key used to open the commitment is sent through an untappable anonymous channel, the value given by the voter to the timeliness member T can be different from the one she provides to the coercer. Hence, the voter who forged the commitment, provides to the coercer the one allowing the coercer to retrieve the vote c, whereas she sends to T the one allowing her to cast the vote a.

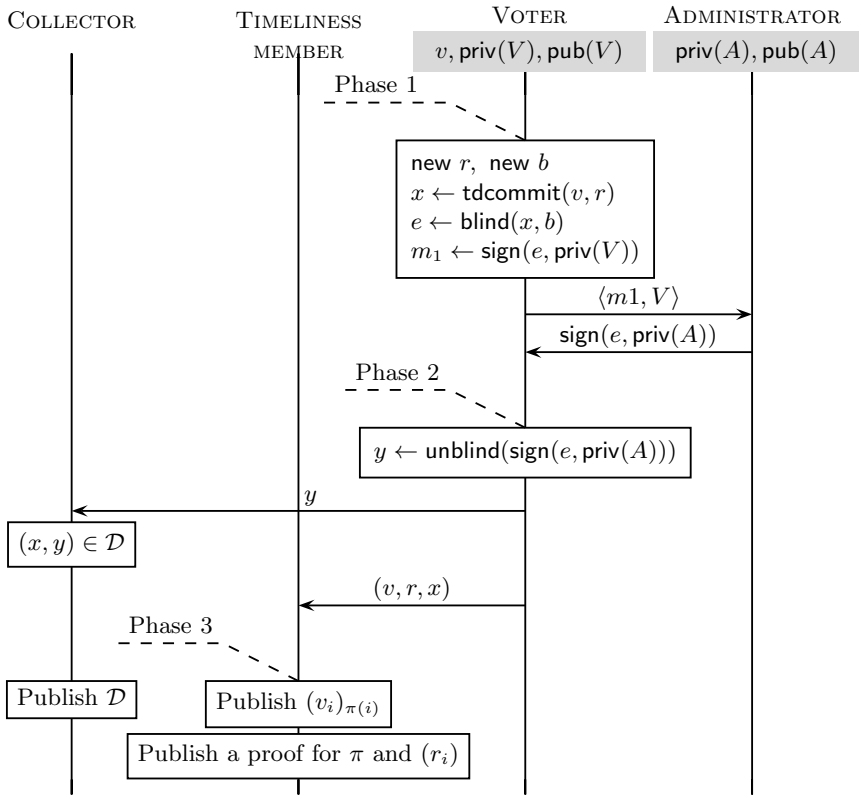

Fig. 2: Okamoto protocol

Coercion-resistance. This scheme is actually not coercion-resistant. If the coercer provides the coerced voter with the commitment that he has to use but without revealing the trap-door, the voter cannot cast her own vote a since the voter cannot produce fake outputs as she did for receipt-freeness.

2 Some RFID protocols

In this section, we describe several RFID protocols following the description proposed in [12]. All these protocols have in common to rely on the exclusive or operator. This operator enjoys some properties that have to be taken into account during the privacy analysis. For this reason, these protocols are out of scope of the existing verification tools such as AKISS [4], APTE [5] or SPEC [11]. All these protocols are quite similar, thus we only describe two of them, namely

KCL'07 and LAK'06, but some other protocols, e.g. OTYT'06, LD'07, YPL'05, *etc* could be used to evaluate the verification algorithms developed during the project.

KCL'07. The reader and the tag id share the secret key k. The reader starts by sending a nonce r_1. The tag generates another nonce r_2 and computes the message $t_0 = \langle id \oplus r_2, \mathsf{h}(\langle r_1, k\rangle) \oplus r_2\rangle$. When receiving such a message, the reader will be able to retrieve r_2, and by xoring it with the second component of the message he received, he will obtained the message $\mathsf{h}(\langle r_1, k\rangle)$.

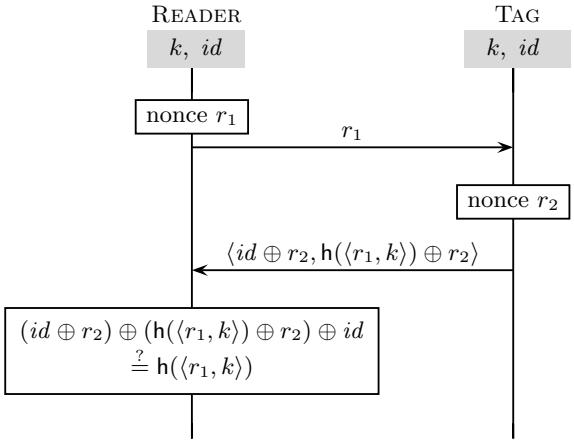

Fig. 3: KCL'07 protocol

The aim of this protocol is not only to authenticate the tag but also to ensure its untraceability. An attacker should not be able to observe whether he has seen the same tag twice or two different tags. An attacker may actually trace the tag by sending a constant r_c instead of a nonce r_1. He would then be able to compute $\mathsf{h}(\langle r_c, k\rangle)$ by xoring id and the two components of the tag's response. Since r_c is a constant, this message is an unique attribute of the tag.

LAK'06. We now present a different protocol [8] which does not suffer from this attack. The main idea is to ask the tag to generate a nonce and to use it to send a different message for each session. As a consequence of this scheme, the tag and the reader must derive new keys after completing a session. We suppose that initially, each tag has his own key k and the reader maintains a database containing those keys. The reader must store for each tag two keys k and k_0 corresponding to the two last keys. Indeed, if the tag has completed entirely the

last session then he will use the key k and k_0 otherwise. The full protocol is given in Figure 2.

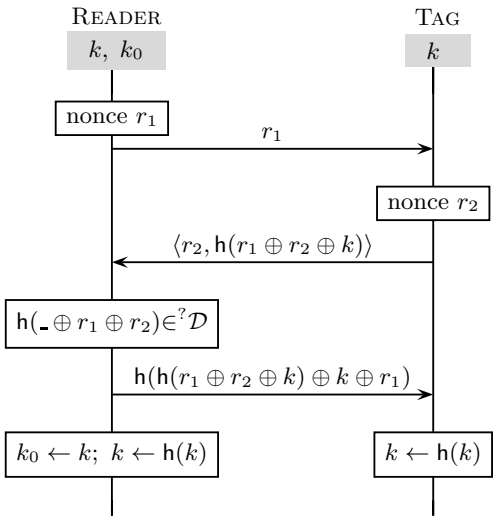

Fig. 4: LAK'06 protocol

The protocol starts similarly to KCL'07. Next, the tag sends $\langle r_2, \mathsf{h}(r_1 \oplus r_2 \oplus k)\rangle$. The reader is then able to compute the function $\mathrm{g}(k') = k' \oplus r_1 \oplus r_2$ and so recognize the tag in the database either with the key k or k_0. He can therefore sends back $\mathsf{h}(\mathsf{h}(r_1 \oplus r_2 \oplus k) \oplus k \oplus r_1)$ to the tag who is now assured that he has been correctly authenticated. In that case he derives his new key $k \leftarrow \mathsf{h}(k)$ from the previous one. So do the reader. The previous attack defeats in this protocol since the attacker could only learn the messages $\mathsf{h}(r_c \oplus r_2 \oplus k)$ which are different for each session (r_2 is a nonce generated by the tag).

Remark that if the tag and the reader kept the same keys then a replay attack would break the authentication of the tag. An attacker could learn r_1 and $\mathsf{h}(r_1 \oplus r_2 \oplus k)$ from a previous complete session. He would then respond to a challenge r_1' by sending back $\langle r_1' \oplus r_1 \oplus r_2, \mathsf{h}(r_1 \oplus r_2 \oplus k)\rangle$. The tag would then authenticate the attacker. Nevertheless, this protocol suffers from another attack on the authentication of the reader [12] (man-in-the-middle attack).

3 E-passport application

An e-passport is a paper passport with an RFID chip that stores the critical information printed on the passport. The International Civil Aviation Organization (ICAO) standard [1] specifies the communication protocols that are used to access these information. We do not plan to describe all the protocols that are specified in the standard but only some of them.

The information stored in the chip is organized in data groups (dg_1 to dg_{19}). For example, dg_5 contains a JPEG copy of the displayed picture, and dg_7 contains the displayed signature. The verification key $\mathsf{vk}(sk_P)$ of the passport, together with its certificate $\mathsf{sign}(\mathsf{vk}(sk_P), sk_{DS})$ issued by the Document Signer authority are stored in dg_{15}. The corresponding signing key sk_P is stored in a tamper resistant memory, and cannot be read or copied. For authentication purposes, a hash of all the dgs together with a signature on this hash value issued by the Document Signer authority are stored in a separate file, the Security Object Document:

$$sod \stackrel{\mathsf{def}}{=} \langle \mathsf{sign}(\mathsf{h}(dg_1, \ldots, dg_{19}), sk_{DS}),\ \mathsf{h}(dg_1, \ldots, dg_{19}) \rangle.$$

The ICAO standard specifies several protocols through which these information can be accessed. First, the Basic Access Control (BAC) protocol establishes a key seed $xkseed$ from which two sessions keys $xksenc$ and $xksmac$ are derived. The purpose of $xksenc$ and $xksmac$ is to prevent skimming and eavesdropping on the subsequent communication with the e-passport. The BAC protocol is given in Figure 3. Once the BAC protocol has been successfully executed, the reader gains access to the information stored in the RFID tag through the Passive Authentication and the Active Authentication protocols that can be executed in any order.

The Basic Access Control (BAC) protocol is a session key establishment protocol. Through BAC, the reader and the passport agree on a key seed $xkseed$ that is then used to generate an encryption session key ($xksenc \leftarrow \mathsf{ekg}(xkseed)$) as well as a mac session key ($xksmac \leftarrow \mathsf{mkg}(xkseed)$). These two session keys are used to secure and provide integrity of subsequent communications. In particular, they are used to encrypt and mac the messages exchanged during the execution of the PA and the AA protocols. The security of the BAC protocol relies on two master keys, ke and km, which are optically retrieved from the passport by the reader before executing the BAC protocol.

It should be noted that some implementations of the BAC protocol breaches unlinkability. For example, in the french implementation, the passport tag replies different error messages depending on whether the nonce in $xenc$ is not n_T or $yenc$ is not a correct mac w.r.t. mac key km [6]. An attacker who does not know the private keys ke and km could then trace a passport in the following way: (i) he first eavesdrop a first session between an authentic reader and a tag T and store $m = \langle xenc, yenc \rangle$; (ii) in a different session, he sends the message m and wait for the tag's response; (iii.a) if he receives a nonce error then he knows that

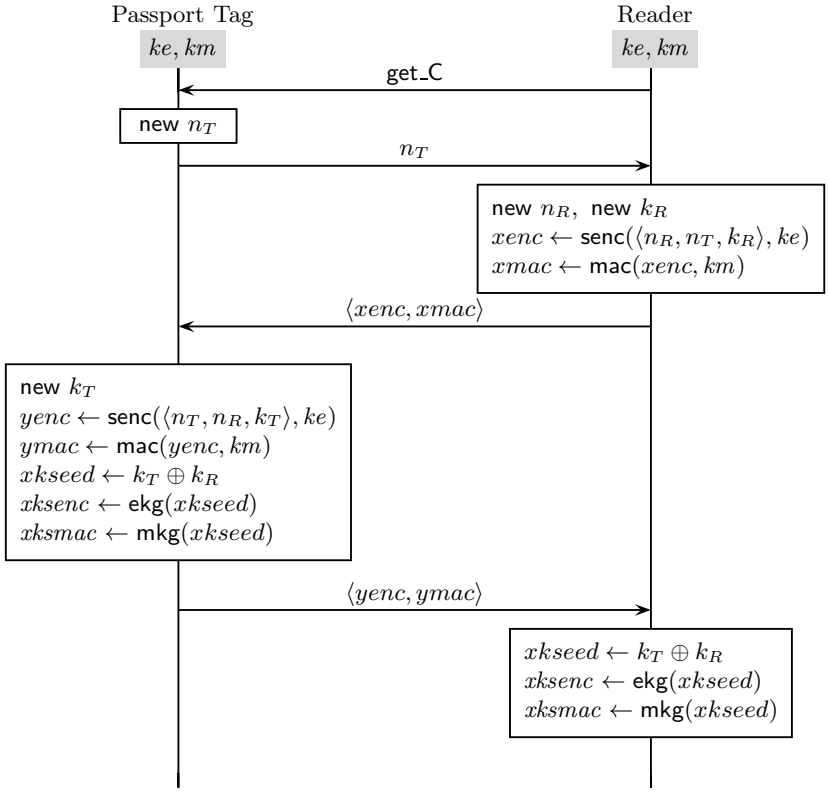

Fig. 5: Basic Access Control protocol

the tag succeeded to mac $xenc$ with his own key ke and so this tag is T; (iii.b) if he receives a MAC error then he knows that the tag is not T. To avoid the information leakage of these error messages, a correct implementation should reply in case of failure a message indistinguishable from $\langle yenc, ymac\rangle$ for the attacker (e.g. $\langle n_1, n_2\rangle$ where n_1, n_2 are fresh nonces).

The Passive Authentication (PA) protocol is an authentication mechanism that proves that the content of the RFID chip is authentic (see Figure 6). Through *PA* the reader retrieves the information stored in the *dg*s and the *sod*. It then verifies that the hash value stored in the *sod* corresponds to the one signed by the Document Signer authority. It further checks that this hash value is consistent with the received *dg*s.

The Active Authentication (AA) protocol is an authentication mechanism that prevents cloning of the passport chip (see Figure 7). It relies on the fact

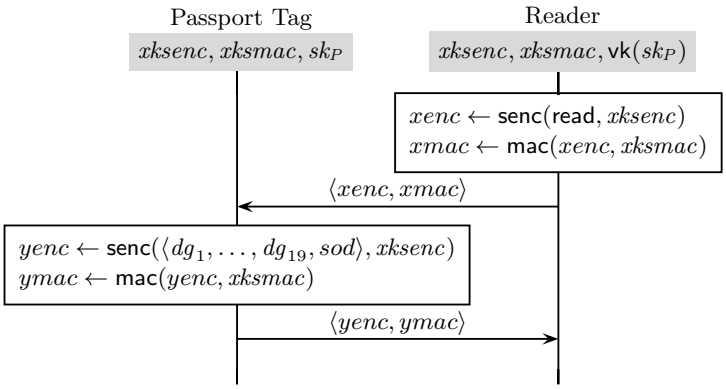

Fig. 6: Passive Authentication protocols

that the secret key sk_P of the passport cannot be read or copied. The reader sends a random challenge to the passport, that has to return a signature on this challenge using its private signature key sk_P. The reader can then verify using the verification key $\mathsf{vk}(sk_P)$ that the signature was built using the expected passport key.

If one wants to prove that e-passport ensures a security property he has to check the whole protocol made of these several smaller sub-protocols. Since the existing tools scale badly, it would be easier to check each sub-protocol separately and prove that it is sufficient to ensure the desired property for the e-passport. This could be achieved by a generic result of composition. This case study will be useful to develop and validate some composition results.

4 3G mobile phones

One of the services offered to 3G telecommunication users is the possibility to send SMSs. The SMSs exchanged between phones (or Mobile Stations - MS) should remain confidential from third parties, and are thus sent encrypted using the *sSMS* protocol. Each sent SMS is encrypted with a different ciphering session key *CK* shared between the network (or Service Network - SN) and the emitting MS, and established through the execution of the *AKA* protocol. We briefly describe these two protocols below.

The AKA protocol. It achieves mutual authentication between a Mobile Station (MS) and the network, and allows them to establish shared session keys to be used to secure subsequent communications. The *AKA* protocol relies on a secret long-term key, K_{IMSI}, shared by the MS with identity *IMSI* and the network.

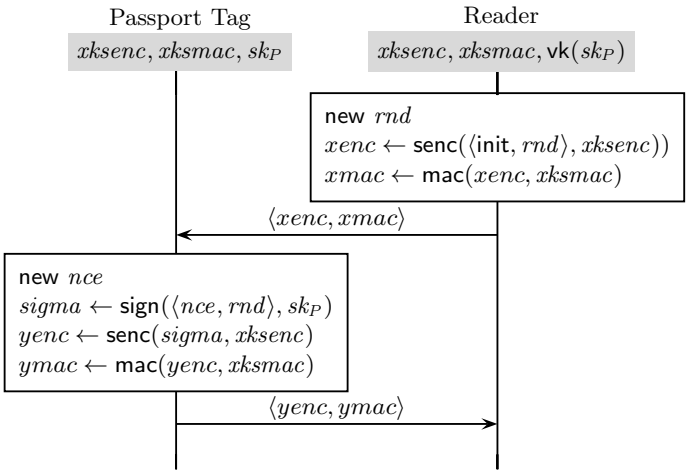

Fig. 7: Active Authentication protocol

This secret long-term key, K_{IMSI}, is assigned to the subscriber by the mobile operator and stored in the USIM.

The *AKA* protocol (see Figure 8) consists in the exchange of two messages: the *authentication request* and the *authentication response*. Before sending an authentication request to the MS, the network computes a fresh random challenge $RAND$, and the authentication token $AUTN$. The functions $\mathsf{f1}-\mathsf{f5}$, used to compute the authentication parameters, are one-way keyed cryptographic functions, and $\oplus$ denotes the exclusive-or operator. $AUTN$ contains a MAC of the concatenation of the random number with a sequence number SQN_N generated by the network using an individual counter for each subscriber. A new sequence number is generated either by increment of the counter or through time based algorithms as defined in [2]. The sequence number SQN_N allows the mobile station to verify the freshness of the authentication request to defend against replay attacks.

The mobile station extracts SQN_N from $AUTN$ and checks that: (i) MAC is a correct mac message w.r.t. K_{IMSI} and replies Mac_{fail} if it is not the case; (ii) the authentication request is fresh (i.e. $SQN_{MS} < SQN_N$) and replies $Sync_{fail}$ otherwise. If the authentication request is correct (i.e. φ_{test} holds) then the mobile station computes the ciphering key CK and stores them in the assignment variable xCK. It also computes the authentication response RES and sends it to the network. The network authenticates the mobile station by verifying whether the received response is equal to the expected one. If so, the network also computes the keys CK and stores it in xCK'.

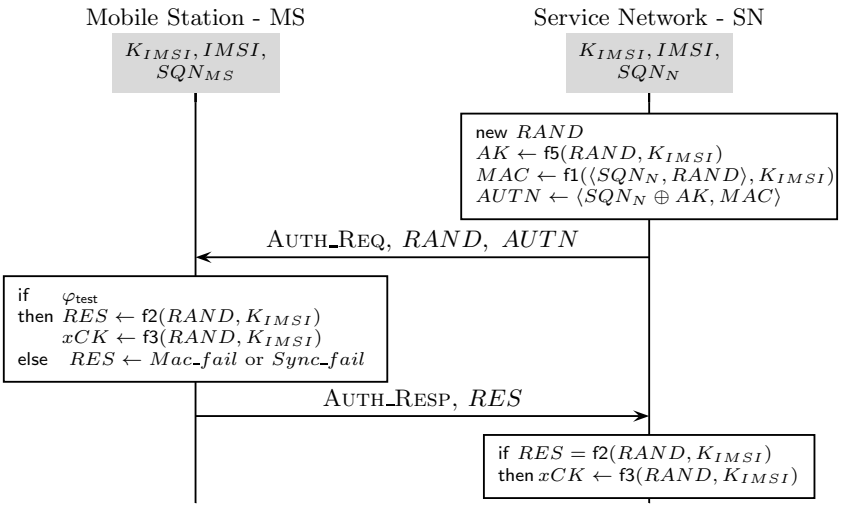

Fig. 8: The AKA protocol

The AKA protocol as deployed in real 3G telecommunication systems presents a linkability attack [3] due to the particular error messages it sends. This attack is very similar to the linkability attack on french e-passport described in section 3. We describe also the fixed version of the AKA protocol proposed in [3] which relies on a public key infrastructure. In particular, in case of failure, i.e. φ_{test} is not satisfied, the answer RES sent by the MS to the SN is encrypted using the public key of the SN, i.e. $\mathsf{pk}(\mathsf{sk}_{SN})$. The fixed AKA protocol is given in Figure 9.

The sSMS protocol. Once the AKA protocol succeeded and established the two session keys xCK and xIK (derived from xCK), the *sSMS* protocol (see Figure 10) allows a MS to send an SMS to another MS through the Network.

The confidentiality of the sent SMS relies on the session key CK established through the execution of the AKA protocol between the MS and the network. The key CK is used to encrypt the two messages exchanged during the execution of the *sSMS* procedure.

The is always the MS that initiates the *sSMS* procedure. It does so by encrypting (using a randomized encryption scheme) the content of the SMS it wants to submit, together with the number of the destination MS, with the session key CK previously established. The message also contains a constant Submit. The Network acknowledges the receipt of this message with a reply containing the time of reception of the submitted SMS encrypted with CK.

Formal methods have been used to prove that unlinkability holds in the fixed version of the AKA protocol [3]. But the exclusive-or operator has been modeled by randomized symmetric encryption since the chosen tool (ProVerif) cannot

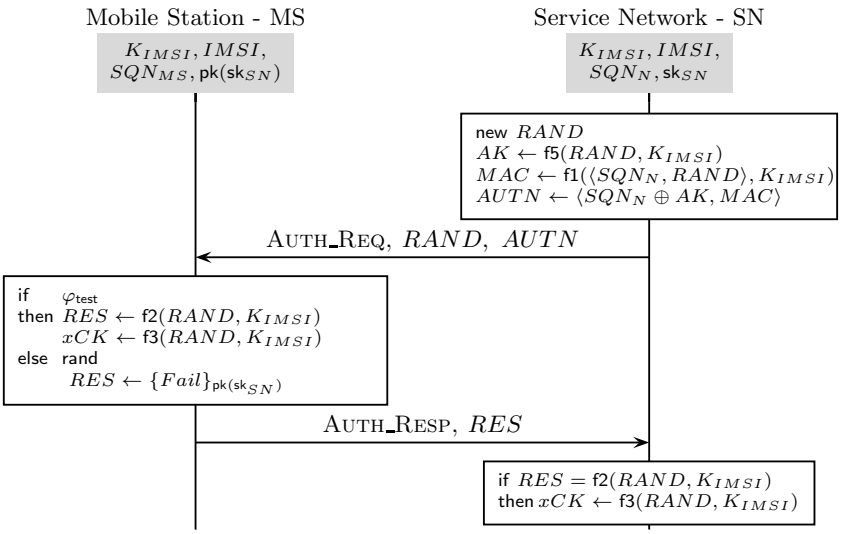

Fig. 9: The *AKA* protocol (variant proposed in [3])

deal with its algebraic properties. Moreover, a compositional result could here be helpful. It would suffice to verify the AKA and the *sSMS* protocols separately in order to make sure that the desired property holds for the whole protocol.

5 Conclusion

This report describes several case studies that we will use as a guideline for our research agenda in the VIP project. Each case study presents its own challenges. In particular, RFID protocols often rely on the exclusive or operator that is out of the scope of the existing verification tools. The two real case studies are interesting since they are made of several subprotocols. This will allow us to guide

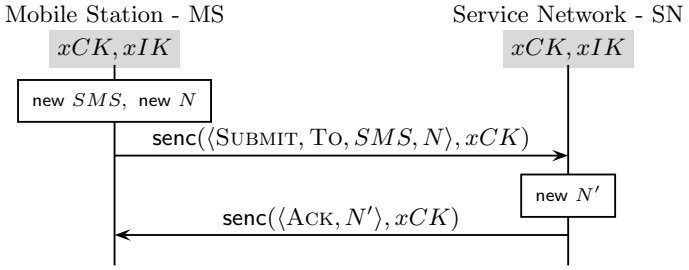

Fig. 10: The *sSMS* protocol

our research about composition. Moreover, they require a careful modelling of the error messages to provide a useful privacy analysis.

References

1. PKI for machine readable travel documents offering ICC read-only access. Technical report, International Civil Aviation Organization, 2004.
2. 3GPP. Technical specification group services and system aspects; 3G security; security architecture (release 9). Technical Report TS 33.102 V9.3.0, 3rd Generation Partnership Project, 2010.
3. M. Arapinis, L. I. Mancini, E. Ritter, M. Ryan, N. Golde, K. Redon, and R. Borgaonkar. New privacy issues in mobile telephony: fix and verification. In *ACM Conference on Computer and Communications Security*, pages 205–216, 2012.
4. R. Chadha, Ş. Ciobâcă, and S. Kremer. Automated verification of equivalence properties of cryptographic protocols. In H. Seidl, editor, *Programming Languages and Systems —Proceedings of the 21th European Symposium on Programming (ESOP'12)*, volume 7211 of *Lecture Notes in Computer Science*, pages 108–127, Tallinn, Estonia, Mar. 2012. Springer.
5. V. Cheval, H. Comon-Lundh, and S. Delaune. Automating security analysis: symbolic equivalence of constraint systems. In J. Giesl and R. Haehnle, editors, *Proceedings of the 5th International Joint Conference on Automated Reasoning (IJCAR'10)*, volume 6173 of *Lecture Notes in Artificial Intelligence*, pages 412–426, Edinburgh, Scotland, UK, July 2010. Springer-Verlag.
6. T. Chothia and V. Smirnov. A traceability attack against e-passports. In *Financial Cryptography and Data Security*, pages 20–34. Springer, 2010.
7. A. Fujioka, T. Okamoto, and K. Ohta. A practical secret voting scheme for large scale elections. In J. Seberry and Y. Zheng, editors, *Advances in Cryptology — AUSCRYPT '92*, volume 718 of *lncs*, pages 244–251. Springer Verlag, 1992.
8. S. Lee, T. Asano, and K. Kim. Rfid mutual authentication scheme based on synchronized secret information. In *Symposium on cryptography and information security*, 2006.
9. T. Okamoto. An electronic voting scheme. In N. Terashima and E. Altman, editors, *Advanced IT Tools*, IFIP The International Federation for Information Processing, pages 21–30. Springer US, 1996.
10. T. Okamoto. Receipt-free electronic voting schemes for large scale elections. In *Proc. 5th Int. Security Protocols Workshop*, volume 1361 of *lncs*. Springer, 1997.
11. A. Tiu and J. Dawson. Automating open bisimulation checking for the spi calculus. In *Computer Security Foundations Symposium (CSF), 2010 23rd IEEE*, pages 307–321. IEEE, 2010.
12. T. van Deursen and S. Radomirovic. Attacks on rfid protocols. *IACR Cryptology ePrint Archive*, 2008:310, 2008.