Security Evaluation of the PLAID Protocol using the ProVerif Tool

Hideki Sakurada
NTT Communication Science Laboratories, NTT Corporation

September 4, 2013

Abstract

The PLAID protocol is a mutual authentication protocol between an integrated circuit card (ICC) and an interface device (IFD). In this report, we analyze the security of PLAID version 8.0. In particular, we analyze secrecy of sensitive data and agreement on keys and some parameters.

1 Protocol Specification

1.1 Abstract

PLAID (Protocol for Light weight Authentication of ID) is a mutual authentication protocol between an integrated circuit card (ICC) an interface device (IFD).

1.2 Basic Reference

We refer to [1] as the specification of the PLAID protocol version 8.0.

1.3 Message Sequence Chart

A message sequence chart (Figure 1 of [1]) of PLAID is shown in Figure 1

The messages consist of terms explained in Table 2 of [1] as shown below.

ACSRecord An Access Control System record for each supported Operational Mode Identifier for the purpose of authorization by back office PACS or LACS access control systems. This record is mapped by the OpModeID to the particular back office numbering system the protocol is supporting. This record is returned by the Final Authenticate command response.

DivData A number (or salt) which is set at PLAID instantiation for use in the key diversification algorithm to ensure that loss of an individual card symmetric key cannot result in a breach of the system

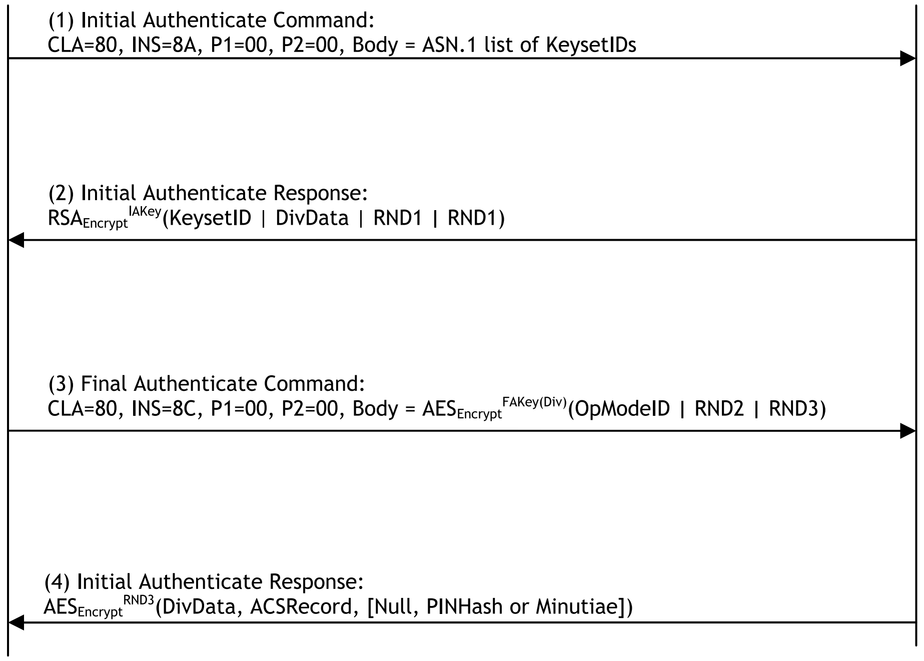
(1) Initial Authenticate Command:
CLA=80, INS=8A, P1=00, P2=00, Body = ASN.1 list of KeysetIDs
(2) Initial Authenticate Response:
$RSA_{Encrypt}^{IAKey}$(KeysetID | DivData | RND1 | RND1)
(3) Final Authenticate Command:
CLA=80, INS=8C, P1=00, P2=00, Body = $AES_{Encrypt}^{FAKey(Div)}$(OpModeID | RND2 | RND3)
(4) Initial Authenticate Response:
$AES_{Encrypt}^{RND3}$(DivData, ACSRecord, [Null, PINHash or Minutiae])

Figure 1: Overview of PLAID 8.0

master keys. This salt is determined by the issuer and should preferably be both random and unique per PLAID invocation AND per system.

FAKey An instance of a Final Authenticate key that is yet to be diversified against an ICC's diversification data. There are only 3 distinct key sizes allowable by AES.

FAKey(DIV) An instance of a Final Authenticate key that Authenticate Key has been diversified against an ICC's (AES) diversification data. There are only 3 distinct key sizes allowable by AES.

KeySetID One or more two byte identifiers sent in a list keyset to the ICC in the Initial Authenticate command so as to determine and/or negotiate the key set to be used for authentication.

Minutiae Minutiae template is extracted as raw data and evaluated by the IFD. At this version we are looking to understand if this is sufficient data for operational systems. We are explicitly seeking comment as to whether additional minutiae data should be designed into the specification or whether minutiae should be by individual finger etc.

OpModeID An identifier sent to the ICC in the Final Authenticate command that determines which ACSRecord record is served up in the final authentication response from the ICC.

PIN The PIN Global to the ICC.

PINHash The SHA1 hash value of the PIN which is served up in the final authentication response from the ICC.

RND1 Random number generated by the smartcard using its TRNG.

RND2 Random number generated by the IFD or back office system using a TRNG.

RND3 String generated by the IFD and ICC separately calculating SHA[RND1—RND2].

1.4 Claimed Security Properties

PLAID is claimed to be highly resilient to the following threats:

ID-leakage The leakage of individually identifiable, unique or determinable data or characteristic of the smartcard or card holder during authentication.

Private-data-leakage Availability of private data in the clear at interfaces accessible by other than the data owner or appropriately authorised parties.

Replay attack An attack in which a valid data transmission from a smartcard is able to be repeated by a different smartcard or by a smartcard emulator and appear to be an authentic session.

Reflection attack An attack where a host can be fooled into accepting a challenge as valid, where the challenge was previously generated by the host in a previous authentication.

Man-in-the-middle attack An attack where an active emulator or similar device or devices insert themselves in the session between the real smartcard and the reader and maliciously modify data within the session in such a fashion that neither the smartcard nor reader detect the modified session.

1.5 Expected Adversary

The specification of PLAID does not explicitly describe the adversary. To capture attacks explained above, we should assume that the adversary can capture messages between ICCs and IFDs, construct messages from captured messages, and send them to ICCs and/or IFDs. It is reasonable to assume cryptographic algorithms, which are public-key and symmetric-key encryption schemes and hash functions, are secure.

1.6 Known Evaluation Results

To our best knowledge, no evaluation result is published while [1] claims that Centerlink has structured a program to "Have PLAID evaluated by the most respected cryptographic organisations, as well as the broader cryptographic community."

2 Security Evaluation by the ProVerif Tool

In this section, we present a brief formal security evaluation of PLAID by the ProVerif tool.

2.1 Evaluation Tool

ProVerif [2] is a tool for automatic analysis of security protocols. We choose ProVerif version 1.86pl4 as an evaluation tool. Please refer to the web page of ProVerif for the technical background.

2.2 Evaluation Level

The protocol assurance level (PAL) in ISO/IEC 29128 [3]. of our evaluation is PAL4. No bounds are put on the number of runs and the maximum size of messages.

2.3 Protocol Model in ProVerif's Specification Language

```
channel c.

type ekey.
type dkey.
type skey.

const null: bitstring.

const KeySetID0: bitstring.
const ICC_ID0: bitstring.
const OpModeID0: bitstring.

fun SHA(bitstring, bitstring): bitstring.

fun rsaencrypt(bitstring, ekey): bitstring.
fun ek(dkey): ekey.
reduc forall m: bitstring, dk: dkey;
      rsadecrypt(rsaencrypt(m, ek(dk)), dk) = m.
fun aesencrypt(bitstring, bitstring): bitstring.
reduc forall m: bitstring, sk: bitstring;
      aesdecrypt(aesencrypt(m, sk), sk) = m.

fun DivDataFun(bitstring): bitstring [private].

table IAKeyTable(bitstring, ekey, dkey).
table FAKeyTable(bitstring, bitstring).
table ACSRecordTable(bitstring, bitstring, bitstring).

free secretIFD: bitstring [private].
free secretICC: bitstring [private].
free ACSRecord0: bitstring [private].

event end().
event beginICC(bitstring, bitstring, bitstring).
event beginIFD(bitstring, bitstring, bitstring).
event endICC(bitstring, bitstring, bitstring).
event endIFD(bitstring, bitstring, bitstring).

query attacker(ACSRecord0).
query attacker(secretIFD).
query attacker(secretICC).
query x: bitstring, y: bitstring, z: bitstring;
  inj-event(endIFD(x, y, z)) ==> inj-event(beginICC(x, y,z )).
query x: bitstring, y: bitstring, z: bitstring;
  inj-event(endICC(x, y, z)) ==> inj-event(beginIFD(x, y, z)).

let IFD =
    in(c, (KeySetID: bitstring, OpModeID: bitstring));
    (* 1) IFD sends the Initial Authentication command *)
    out(c, KeySetID);
    (* 3) The IFD repsonds to the IA repsonse *)
    in (c, ESTR1: bitstring);
    get IAKeyTable(=KeySetID, IAKey_e, IAKey_d) in
    let STR1 = rsadecrypt(ESTR1, IAKey_d) in
    let (=KeySetID, DivData: bitstring, RND1: bitstring, =RND1)
        = STR1 in
    new RND2: bitstring;
    let RND3 = SHA(RND1, RND2) in
    event beginIFD(KeySetID, OpModeID, RND3);
    get FAKeyTable(=KeySetID, FAKey) in
    let FAKey_Div = aesencrypt(DivData, FAKey) in
    let STR2 = (OpModeID, RND2, RND3) in
    let ESTR2 = aesencrypt(STR2, FAKey_Div) in
    out(c, ESTR2);
    (* 7) The IFD processes the credential *)
    in (c, ESTR3: bitstring);
    let (=DivData, ACSRecord: bitstring, auth: bitstring)
       = aesdecrypt(ESTR3, RND3) in
    (* Security *)
    event endIFD(OpModeID, ACSRecord, RND3);
    out(c, aesencrypt(secretIFD, RND3));
    event end().

let ICC =
    in(c, ICC_ID: bitstring);
    (* 2) ICC responds to the IA command *)
    in(c, KeySetID: bitstring);
    get IAKeyTable(=KeySetID, IAKey_e, IAKey_d) in
    new RND1: bitstring;
    let DivData = DivDataFun(ICC_ID) in
    let STR1 = (KeySetID, DivData, RND1, RND1) in
    let ESTR1 = rsaencrypt(STR1, IAKey_e) in
    out(c, ESTR1);
    (* 4) The ICC repsonds to the FA command *)
    in(c, ESTR2: bitstring);
    get FAKeyTable(=KeySetID, FAKey) in
    let FAKey_Div = aesencrypt(DivData, FAKey) in
    let STR2 = aesdecrypt(ESTR2, FAKey_Div) in
    let (OpModeID: bitstring, RND2: bitstring, RND3: bitstring)
        = STR2 in
    if RND3 = SHA(RND1, RND2) then
    get ACSRecordTable(=ICC_ID, =OpModeID, ACSRecord) in
    event beginICC(OpModeID, ACSRecord, RND3);
    let STR3 = (DivData, ACSRecord, null) in
    let ESTR3 = aesencrypt(STR3, RND3) in
    out(c, ESTR3);
    (* Security *)
    event endICC(KeySetID, OpModeID, RND3);
    out(c, aesencrypt(secretICC, RND3));
    event end().

process
new IAKey_d0: dkey;
let IAKey_e0 = ek(IAKey_d0) in
insert IAKeyTable(KeySetID0, IAKey_e0, IAKey_d0);
new FAKey0: bitstring;
insert FAKeyTable(KeySetID0, FAKey0);
insert ACSRecordTable(ICC_ID0, OpModeID0, ACSRecord0);
(
(!IFD) | (!ICC)
)
```

2.4 Adversarial Model

ProVerif assumes Dolev-Yao network model. It is suitable for the evaluation of PLAID because it can capture all attacks PLAID to which PLAID is claimed to be resilient.

2.5 Security Properties Description

Security properties are also specified in the protocol model above. In our evaluation, secrecy of ACSRecord and RND3 as well as mutual authentication are evaluated.

In our evaluation, we do not evaluate secrecy of identities under ID-leakage attacks because it requires us more efforts.

Secrecy of ACSRecord and RND3 are specified as the following queries:

```
query attacker(ACSRecord0).
query attacker(secretIFD).
query attacker(secretICC).
```

Since RND3 is freshly generated in each execution, secrecy of RND3 cannot be specified as a signle query. Instead, we let an IFD and an ICC to send constants secretIFD and secretICC encrypted by RND3, respectively, and evaluate secrecy of these constants.

Mutual authentication is specified as the following queries:

```
query x: bitstring, y: bitstring, z: bitstring;
  inj-event(endIFD(x, y, z)) ==> inj-event(beginICC(x, y, z)).
query x: bitstring, y: bitstring, z: bitstring;
  inj-event(endICC(x, y, z)) ==> inj-event(beginIFD(x, y, z)).
```

The first query evaluates authentication of an ICC by an IFD: if an IFD finishes with some values of OpModeID, ACSRecord, and RND3, there is an ICC that executes the protocol with these values. The second query evaluates authentication of an IFD by an ICC: if an IFD finishes with some values of KeySetID, ACSRecord, and RND3, there is an IFD that executes the protocol with these values.

2.6 Evaluation Results

ProVerif shows that PLAID has all security properties above. The output of ProVerif is summarized as follows.

```
RESULT inj-event(endICC(x,y,z))
       ==> inj-event(beginIFD(x,y,z)) is true.
RESULT inj-event(endIFD(x_1010,y_1011,z_1012))
       ==> inj-event(beginICC(x_1010,y_1011,z_1012)) is true.
RESULT not attacker(secretICC[]) is true.
RESULT not attacker(secretIFD[]) is true.
RESULT not attacker(ACSRecord0[]) is true.
```

2.7 Modeling

In our modeling of PLAID, we make the following assumptions:

- Each list of KetSetIDs consists of a signle KeySetID.
- DivData is unique for each ICC.
- Neither PINHash nor Minutiae are used.
- For each KeySetID, there is unique FAKey that are securely shared among all ICCs and IFDs.
- For each KeySetID, there is a unique pair of public IAKey and private IAKey, which is securely stored in IFDs.
- Each entity aborts if it receives an unexpected message.
- All cryptographic primitives including public-key and symmetric-key encryption schemes and hash functions are ideally secure.

2.8 Evaluation Cost

Evaluation Environment

CPU Intel Core i7 L620 (2.00GHz)

RAM 500MB

OS Ubuntu Linux 12.04.2 LTS on Oracle VM VirtualBox 4.1.6 on Microsoft Windows 7 Professional (32-bit).

Time It takes 0.03 seconds.

3 Comments on the specification of PLAID

In the process of our evaluation, we find that it is desirable that the following information is more clearly specified in the specification of the protocol:

- the data initially stored on each ICC, IFD, and the backend access control system,
- the data initially shared between these entities, and
- the data that must be shared or agreed between these entities after an execution of the protocol.

References

[1] Centerlink. Protocol for Lightweight Authentication of Identity (PLAID) - LOGICAL SMARTCARD APPLICATION SPECIFICATION PLAID Version 8.0 - FINAL, December 2009. Available from `http://www.humanservices.gov.au/corporate/publications-and-resources/plaid/`.

[2] Proverif: Cryptographic protocol verifier in the formal model. `http://prosecco.gforge.inria.fr/personal/bblanche/proverif/`.

[3] ISO/IEC 29128:2011, information technology - security techniques - verification of cryptographic protocols, 2009.