Security Analysis of PLAID

Dai Watanabe[1]

Yokoyama Laboratory, Hitachi, Ltd.,
292 Yoshida-cho, Totsuka-ku, Yokohama, 244-0817, Japan
dai.watanabe.td@hitachi.com

Abstract. PLAID is a mutual authentication protocol for a smartcard system. In this report we formally analyse its security by using Scyther tool and prove that PLAID achieves non-injective agreement for both roles (IFD and ICC).

Keywords. PLAID 8.0, cryptographic protocol, formal analysis, Scyther

1 Protocol Specification

1.1 Abstract

PLAID (Protocol for Light weight Authentication of ID) 8.0 is a mutual authentication protocol between an Interface Device (IFD) and an Integrated Circuit Card (ICC). It was proposed by Centrelink in 2009 [2]. We abbreviate the version number of the protocol and it is just denoted by PLAID in the rest of this report.

1.2 Basic Reference

Throughout this report [2] is referred to as the primary source of the specification of PLAID protocol.

1.3 Message Sequence Chart

Figure 1 shows the rough sketch of the message sequence chart of PLAID. The meaning of each term in the chart is given as follows (extracted from [2]):

ACSRecord : An Access Control System record for each supported Operational Mode identifier for the purpose of authentication by back office PACS or LACS access control systems. 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 master keys. This salt is determined by the issuer and should preferably be both random and unique per PLAID invocation and per system.

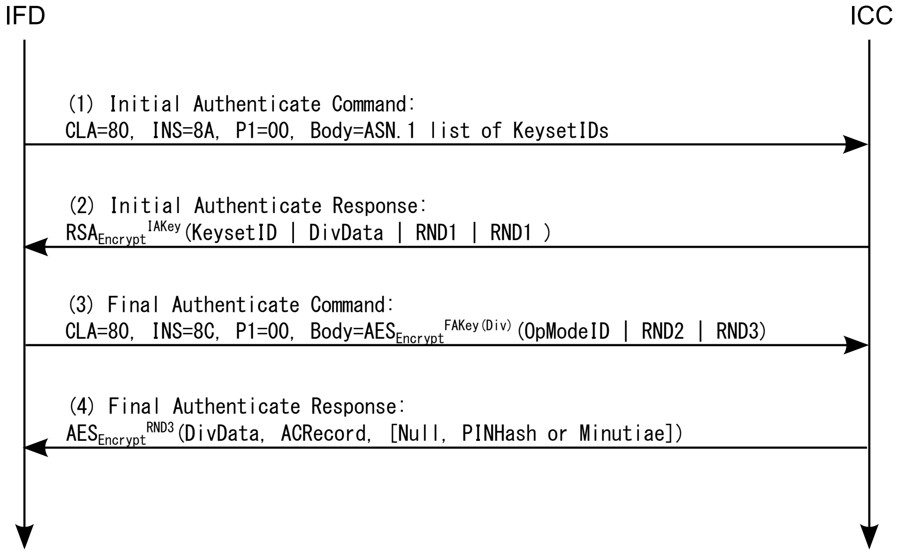
IFD
ICC
(1) Initial Authenticate Command:
CLA=80, INS=8A, P1=00, Body=ASN.1 list of KeysetIDs
(2) Initial Authenticate Response:
$\text{RSA}_{\text{Encrypt}}^{\text{IAKey}}$(KeysetID | DivData | RND1 | RND1)
(3) Final Authenticate Command:
CLA=80, INS=8C, P1=00, Body=$\text{AES}_{\text{Encrypt}}^{\text{FAKey(Div)}}$(OpModeID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

Fig. 1. PLAID 8.0 Authentication Protocol Overview

FAKey : An instance of a Final Authenticate key that is yet to be diversified against an ICC's diversification data.

FAKey(DIV) : An instance of a Final Authenticate key that has been diversified against ICC's diversification data.

KeySetID : One or more identifiers sent in a list 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.

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 SHA1(RND1 — RND2).

RND3 may be used for subsequent communication with the ICC. The encryption functions used in PLAID are SHA-1, AES, and RSA (with PKCS 1.5 or OAEP padding). The key lengths of AES and RSA are variable and they are determined in the first two messages of the protocol.

1.4 Claimed Security Properties

PLAID is claimed to be highly resilient to the following 5 threats in [2]:

ID-leakage : A constant subset of data that is static for each authentication exchange between a specific ICC and IFD. This subset (even when encrypted) could allow for identification of an individual smartcard, and therefore indirectly the cardholder.

Private-data-leakage : The 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 an ICC is able to be repeated by a different ICC or by an ICC emulator and appear to be an authentic session as viewed from an IFD.

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 ICC and the IFD and maliciously modify data within the session in such a fashion that neither the ICC nor IFD delete the modified session.

1.5 Expected Adversary

No specific description on the adversary is found in [2]. On the other hand, PLAID is an authentication protocol between a smartcard and devices and near field wireless communication is expected. In addition, the claimed security properties indicates both active and passive adversaries are expected, i.e., the adversary can eavesdrop, modify, insert packets transmitted between the IFD and the ICC.

1.6 Known Evaluation Results

To our best knowledge, there is no published security evaluation on PLAID 8.0 while [2] claims "*PLAID have evaluated by the most respected cryptographic organizations, as well as the broader cryptographic community*".

2 Security Evaluation by Scyther Tool

In this section, we present a brief formal security evaluation of PLAID by Scyther Tool.

2.1 Evaluation Tool

We chose Scyther v1.1 [3] as as evaluation tool. Scyther is a tool for the automatic verification of security protocols. Please refer to [4] for the technical background of Scyther.

2.2 Evaluation Level

Our model is evaluated by Scyther without any options and it means that the evaluation level corresponds to PAL3 in [1]. The number of runs are restricted to 5 in our evaluation.

2.3 Protocol Model in Scyther Language

```
/****************************************************/
/* constants */
usertype String, Number;
const 808A0000: Number;
const 808C0000: Number;

/****************************************************/
/* key generation function */
hashfunction SHA;
const choice: Function;
const KeysetIDs: Function;
hashfunction FAKey;

/****************************************************/
/* macros */
/*----------------------*/
macro InitialAuthenticateCommand = ( 808A0000, KeysetIDs(IFD) );

/*----------------------*/
macro KeysetID = choice(KeysetIDs, ICC);
macro STR1 = ( KeysetID, DivData, RND1, RND1 );
macro IAKey = pk(IFD);
macro InitialAuthenticateResponse = {STR1}IAKey;

/*----------------------*/
macro OpModeID = KeysetID;
macro RND3 = SHA(RND1, RND2);
macro FAKeyDiv = {DivData}FAKey(KeysetID, k(IFD, ICC));
macro FinalAuthenticateCommand = ( 808C0000, {OpModeID, RND2, RND3}FAKeyDiv );

/*----------------------*/
macro ACSRecord = KeysetID;
macro PINHash = SHA(k(IFD, ICC));
macro FinalAuthenticateResponse = {DivData, ACSRecord, PINHash}RND3;

///////////////////////////////////////////////////////////////////////
protocol @KeySwap(X)
{
  role X
    {
      var IFD, ICC: Agent;

      recv_!x1( X,X, k(IFD,ICC) );
      send_!x2( X,X, k(ICC,IFD) );
    }
}

///////////////////////////////////////////////////////////////////////
protocol PAID(IFD, ICC)
{
  /****************************************************/
  role IFD
  {
    /*----------------------------------------------*/
    //variables
    var DivData: Ticket;
    var RND1: Ticket; //generated by TRNG
    fresh RND2: Nonce; //generated by TRNG

    /*----------------------------------------------*/
    //sequence

    send_1(IFD, ICC, InitialAuthenticateCommand);
    recv_2(ICC, IFD, InitialAuthenticateResponse);
    send_3(IFD, ICC, FinalAuthenticateCommand);
    recv_4(ICC, IFD, FinalAuthenticateResponse);

    /*----------------------------------------------*/
    //security properties
  }

  /****************************************************/
  role ICC
  {
    /*----------------------------------------------*/
    //variables
    fresh DivData: Nonce;
    fresh RND1: Nonce; //generated by TRNG
    var RND2: Ticket; //generated by TRNG

    /*----------------------------------------------*/
    //sequence
    recv_1(IFD, ICC, InitialAuthenticateCommand);
    send_2(ICC, IFD, InitialAuthenticateResponse);
    recv_3(IFD, ICC, FinalAuthenticateCommand);
    send_4(ICC, IFD, FinalAuthenticateResponse);

    /*----------------------------------------------*/
    //security properties
  }
}
```

2.4 Adversarial Model

Scyther assumes Dolev-Yao network model and it is suitable for the evaluation of PLAID. Besides, [2] assumes that users who share a key does not abuse their secret to break the PLAID authentication system.

2.5 Security Properties Description

Security Properties for the IFD Are given as follows:

```
    claim(IFD, Weakagree);
    claim(IFD, Niagree);
    claim(IFD, Nisynch);
    claim(IFD, Secret, k(IFD,ICC));
    claim(IFD, SKR, RND3 );
```

Here the first claim (`Weakagreement` of the IFD) is essential to check if the authentication is successful and the second and the last one is important if `RND3` is used as a session key.

Security Properties for the ICC Are given as follows:

```
    claim(ICC, Weakagree);
    claim(ICC, Niagree);
    claim(ICC, Nisynch);
    claim(ICC, Secret, k(IFD,ICC));
    claim(ICC, SKR, RND3 );
```

Here the first claim (`Weakagreement` of the ICC) is essential to check if the authentication is successful and the second and the last one is important if `RND3` is used as a session key.

2.6 Evaluation Results

Output of the Tool Table 1 shows the summary of Scyther's output. We can see that most of the claimed security properties are satisfied, but non-injective synchronization is not satisfied.

Attack Graph and Its Explanation Figure 2 gives an example in which the non-injective synchronization for the ICC role is not satisfied. By intuition, the reason of this vulnerability is that Initial Authenticate command is considered

Table 1. Evaluation result by Scyther

claim	PAID,IFD	Weakagree_IFD1	--	Ok	[no attack within bounds]
claim	PAID,IFD	Niagree_IFD2	--	Ok	[no attack within bounds]
claim	PAID,IFD	Nisynch_IFD3	--	Fail	[at least 2 attacks]
claim	PAID,IFD	Secret_IFD4	k(IFD,ICC)	Ok	[no attack within bounds]
claim	PAID,IFD	SKR_IFD5	SHA(RND1,RND2)	Ok	[no attack within bounds]
claim	PAID,ICC	Weakagree_ICC1	--	Ok	[no attack within bounds]
claim	PAID,ICC	Niagree_ICC2	--	Ok	[no attack within bounds]
claim	PAID,ICC	Nisynch_ICC3	--	Fail	[at least 2 attacks]
claim	PAID,ICC	Secret_ICC4	k(IFD,ICC)	Ok	[no attack within bounds]
claim	PAID,ICC	SKR_ICC5	SHA(RND1,RND2)	Ok	[no attack within bounds]

to be IFD-dependent constant and the adversary can capture and replace it. This does not sound a serious vulnerability, if the system implementation avoids a ciphersuite downgrade attack.

2.7 Modeling

Modeling Process

- A list of KeySetID (KeySetIDs) is modeled as an agent dependent constant. The KeySetID chosen in the second message is modeled as a constant dependent on the IFD and the ICC.
- DivData, RND1, RND2 are modeled as variables of Nonce type.
- The relation between OpModeID and KeySetID is variable in [2]. They are considered that an OpModeId is tied to a KeySetID. ACSRecord is considered in the same manner.
- The CBC mode of operation is chosen in PLAID for block cipher encryption. However, our model does not include an IV in any message because both the 3rd and the 4th messages encrypted by the AES include randomly generated nonces RND2 and RND3 in its payload or in its encryption key. In Scyther model, these terms ensure that the messages are well randomized.
- Scyther model cannot describe a reliable role or a key shared by a plural agents. In our model, an IFD and an ICC share a proper secret key and no other agent knows it.

Validity of the Modeling

2.8 Limitation of Scyther Tool

Here we describe the limitation of our evaluation arising from Scyther's features.

Possibility to Model Protocol Nothing to be written here.

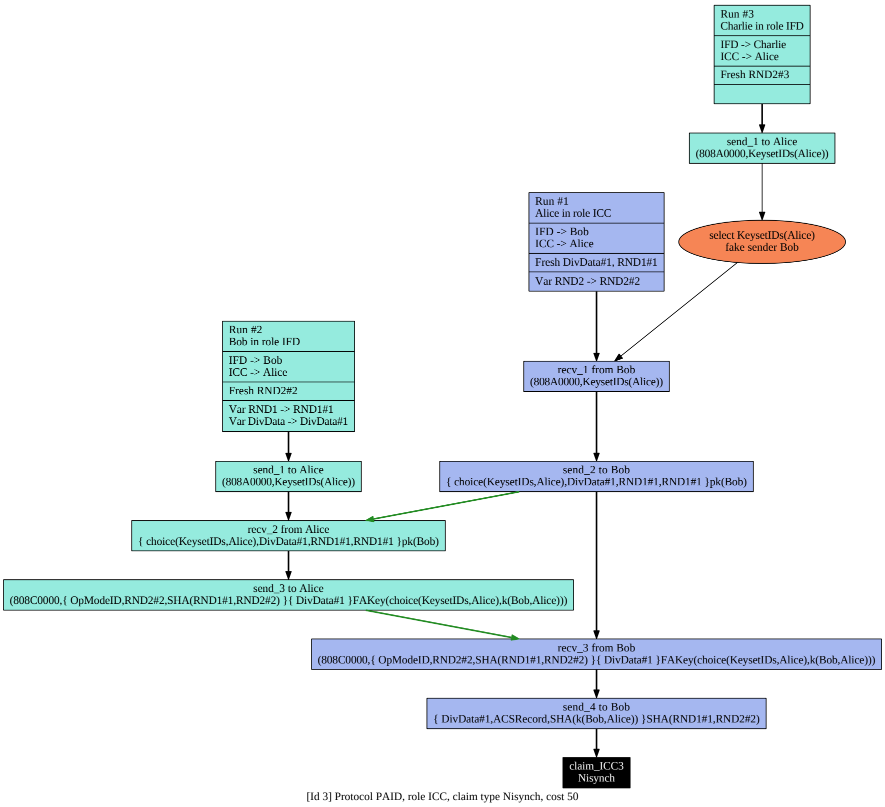
Run #3
Charlie in role IFD
IFD -> Charlie
ICC -> Alice
Fresh RND2#3
send_1 to Alice
(808A0000,KeysetIDs(Alice))
select KeysetIDs(Alice)
fake sender Bob
Run #1
Alice in role ICC
IFD -> Bob
ICC -> Alice
Fresh DivData#1, RND1#1
Var RND2 -> RND2#2
Run #2
Bob in role IFD
IFD -> Bob
ICC -> Alice
Fresh RND2#2
Var RND1 -> RND1#1
Var DivData -> DivData#1
recv_1 from Bob
(808A0000,KeysetIDs(Alice))
send_1 to Alice
(808A0000,KeysetIDs(Alice))
send_2 to Bob
{ choice(KeysetIDs,Alice),DivData#1,RND1#1,RND1#1 }pk(Bob)
recv_2 from Alice
{ choice(KeysetIDs,Alice),DivData#1,RND1#1,RND1#1 }pk(Bob)
send_3 to Alice
(808C0000,{ OpModeID,RND2#2,SHA(RND1#1,RND2#2) }{ DivData#1 }FAKey(choice(KeysetIDs,Alice),k(Bob,Alice)))
recv_3 from Bob
(808C0000,{ OpModeID,RND2#2,SHA(RND1#1,RND2#2) }{ DivData#1 }FAKey(choice(KeysetIDs,Alice),k(Bob,Alice)))
send_4 to Bob
{ DivData#1,ACSRecord,SHA(k(Bob,Alice)) }SHA(RND1#1,RND2#2)
claim_ICC3
Nisynch
[Id 3] Protocol PAID, role ICC, claim type Nisynch, cost 50

Fig. 2. An attack graph which explains how non-injective synchronization is broken

Possibility to Model Adversary Nothing to be written here.

Possibility to Describe Security Properties Scyther cannot evaluate *injectivity*, which is required for checking the applicability of replay attack. In addition, Scyther assumes that any role knows the identities of other roles (So no hidden ID can exist in Scyther models). Therefore we cannot evaluate the resilience of PLAID against *ID-leakage*.

2.9 Evaluation Cost

Evaluation Environment

CPU : Intel Core2Duo E8400 (3.0 GHz)
RAM : 4.0 GB
OS : Microsoft Windows 7 Professional (32-bit)

Time It takes 1.9 seconds for the evaluation of the model by Scyther v1.1. Though I tried `--unbounded` option of Scyther, it automatically gave up unbounded evaluation and proceeded bounded one.

References

1. ISO/IEC 29128:2011, Information technology – Security techniques – Verification of cryptographic protocols, 2011.
2. Centrelink, Protocol for Lightweight Authentication of Identity (PLAID)– LOGICAL SMARTCARD APPLICATION SPECIFICATION PLAID Version 8.0 - FINAL, December 2009. Available at `http://www.humanservices.gov.au/corporate/publications-and-resources/plaid/`.
3. Cas Cremers, The Scyther tool, `http://people.inf.ethz.ch/cremersc/scyther/`.
4. Cas Cremers and Sjouke Mauw, "Operational Semantics and Verification of Security Protocols," Springer-Verlag, 2012.