

# Appendix A

## Google's Secure Element API

This appendix lists the interface definition of Google's proprietary secure element API as included in version 4.2.1 of the Android system. See Sect. 6.2.4.1 for a detailed analysis.

### A.1 Class NfcAdapterExtras

```
1  /*
2  * Copyright (C) 2011 The Android Open Source Project
3  *
4  * Licensed under the Apache License, Version 2.0 (the
5  * "License"); you may not use this file except in compliance
6  * with the License. You may obtain a copy of the License at
7  *
8  *     http://www.apache.org/licenses/LICENSE-2.0
9  *
10 * Unless required by applicable law or agreed to in writing,
11 * software distributed under the License is distributed on
12 * an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY
13 * KIND, either express or implied. See the License for the
14 * specific language governing permissions and limitations
15 * under the License.
16 */
17
18 package com.android.nfc_extras;
19
20 public final class NfcAdapterExtras {
21     /**
22      * Broadcast Action: RF field ON has been detected.
23      * This is an unreliable signal, and will be removed.
24      */
25     public static final String ACTION_RF_FIELD_ON_DETECTED =
26         "com.android.nfc_extras.action.RF_FIELD_ON_DETECTED";
27
28     /**
29      * Broadcast Action: RF field OFF has been detected.
30      * This is an unreliable signal, and will be removed.
31      */
32     public static final String ACTION_RF_FIELD_OFF_DETECTED =
33         "com.android.nfc_extras.action.RF_FIELD_OFF_DETECTED";
34 }
```

```

34
35
36 /**
37  * Get the NfcAdapterExtras for the given NfcAdapter.
38  */
39 public static NfcAdapterExtras get(NfcAdapter adapter) { ... }
40
41
42 /**
43  * Immutable data class that describes a card emulation route.
44  */
45 public final static class CardEmulationRoute {
46     /**
47      * Card Emulation is turned off on this NfcAdapter.
48      */
49     public static final int ROUTE_OFF = 1;
50
51     /**
52      * Card Emulation is routed to nfcEe only when the screen
53      * is on, otherwise it is turned off.
54      */
55     public static final int ROUTE_ON_WHEN_SCREEN_ON = 2;
56
57     /**
58      * A route such as ROUTE_OFF or ROUTE_ON_WHEN_SCREEN_ON.
59      */
60     public final int route;
61
62     /**
63      * The NfcExecutionEnvironment that Card Emulation is
64      * routed to.
65      */
66     public final NfcExecutionEnvironment nfcEe;
67
68
69     public CardEmulationRoute(
70         int route, NfcExecutionEnvironment nfcEe) { ... }
71 }
72
73
74 /**
75  * Get the current routing state of the secure element.
76  */
77 public CardEmulationRoute getCardEmulationRoute() { ... }
78
79 /**
80  * Set the routing state of the secure element.
81  */
82 public void setCardEmulationRoute(
83     CardEmulationRoute route) { ... }
84
85
86 /**
87  * Get the NfcExecutionEnvironment for the embedded secure
88  * element.
89  */
90 public NfcExecutionEnvironment getEmbeddedExecutionEnvironment(
91     ) { ... }
92
93
94 /**
95  * Authenticate the client application.
96  * Some implementations of NFC Adapter Extras may require
97  * applications to authenticate with a token, before using
98  * other methods.
99  * This method is not used on Nexus S/Galaxy Nexus.
100 */

```

```

101 public void authenticate(byte[] token) { ... }
102
103 /**
104  * Returns the name of this adapter's driver.
105  */
106 public String getDriverName() { ... }
107 }

```

## A.2 Class NfcExecutionEnvironment

```

1  /*
2  * Copyright (C) 2011 The Android Open Source Project
3  *
4  * Licensed under the Apache License, Version 2.0 (the
5  * "License"); you may not use this file except in compliance
6  * with the License. You may obtain a copy of the License at
7  *
8  *     http://www.apache.org/licenses/LICENSE-2.0
9  *
10 * Unless required by applicable law or agreed to in writing,
11 * software distributed under the License is distributed on
12 * an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY
13 * KIND, either express or implied. See the License for the
14 * specific language governing permissions and limitations
15 * under the License.
16 */
17
18 package com.android.nfc_extras;
19
20 import java.io.IOException;
21
22 public class NfcExecutionEnvironment {
23     /**
24      * Broadcast Action: An ISO-DEP AID was selected.
25      */
26     public static final String ACTION_AID_SELECTED =
27         "com.android.nfc_extras.action.AID_SELECTED";
28     /**
29      * Mandatory byte array extra field in ACTION_AID_SELECTED.
30      */
31     public static final String EXTRA_AID =
32         "com.android.nfc_extras.extra.AID";
33
34     /**
35      * Broadcast action: A filtered APDU was received.
36      */
37     public static final String ACTION_APDU_RECEIVED =
38         "com.android.nfc_extras.action.APDU_RECEIVED";
39     /**
40      * Mandatory byte array extra field in ACTION_APDU_RECEIVED.
41      */
42     public static final String EXTRA_APDU_BYTES =
43         "com.android.nfc_extras.extra.APDU_BYTES";
44
45     /**
46      * Broadcast action: An EMV card removal event was detected.
47      */
48     public static final String ACTION_EMV_CARD_REMOVAL =
49         "com.android.nfc_extras.action.EMV_CARD_REMOVAL";
50 }

```

```
51  /**
52   * Broadcast action: An adapter implementing MIFARE Classic
53   * via card emulation detected that a block has been accessed.
54   */
55   public static final String ACTION_MIFARE_ACCESS_DETECTED =
56       "com.android.nfc_extras.action.MIFARE_ACCESS_DETECTED";
57   /**
58   * Optional integer extra field in ACTION_MIFARE_ACCESS
59   * _DETECTED that provides the block number being accessed.
60   */
61   public static final String EXTRA_MIFARE_BLOCK =
62       "com.android.nfc_extras.extra.MIFARE_BLOCK";
63
64
65   /**
66   * Open the NFC Execution Environment on its contact
67   * interface.
68   */
69   public void open() throws IOException { ... }
70
71   /**
72   * Close the NFC Execution Environment on its contact
73   * interface.
74   */
75   public void close() throws IOException { ... }
76
77   /**
78   * Send raw commands to the NFC Execution Environment
79   * and receive the response.
80   */
81   public byte[] transceive(byte[] in) throws IOException { ... }
82 }
```

# Appendix B

## Modifications to Google's Secure Element API Library

This appendix lists a modified version of Google's proprietary secure element API that outputs debug information to the Android debug log. This version is based on `com.android.nfc_extras` as included in Android 4.1.1. Most comments were removed from these listings.

### B.1 Class `NfcAdapterExtras`

```
1  /*
2  * Copyright (C) 2011 The Android Open Source Project
3  * Modifications (debug output): (C) 2012 Michael Roland
4  *
5  * Licensed under the Apache License, Version 2.0 (the
6  * "License"); you may not use this file except in compliance
7  * with the License. You may obtain a copy of the License at
8  *
9  *     http://www.apache.org/licenses/LICENSE-2.0
10 *
11 * Unless required by applicable law or agreed to in writing,
12 * software distributed under the License is distributed on
13 * an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY
14 * KIND, either express or implied. See the License for the
15 * specific language governing permissions and limitations
16 * under the License.
17 */
18
19 package com.android.nfc_extras;
20
21 import java.util.HashMap;
22
23 import android.content.Context;
24 import android.nfc.INfcAdapterExtras;
25 import android.nfc.NfcAdapter;
26 import android.os.RemoteException;
27 import android.util.Log;
28 import java.lang.reflect.Method;
```

```

37 public final class NfcAdapterExtras {
38     private static final String TAG = "NfcAdapterExtras";

48     public static final String ACTION_RF_FIELD_ON_DETECTED =

59     public static final String ACTION_RF_FIELD_OFF_DETECTED =
60         "com.android.nfc_extras.action.RF_FIELD_OFF_DETECTED";

65     private static INfcAdapterExtras sService;
66     private static final CardEmulationRoute ROUTE_OFF =
67         new CardEmulationRoute(CardEmulationRoute.ROUTE_OFF,
68             null);

71     private static final HashMap<NfcAdapter, NfcAdapterExtras>
72         sNfcExtras = new HashMap();
73
74     private final NfcExecutionEnvironment mEmbeddedEe;
75     private final CardEmulationRoute mRouteOnWhenScreenOn;
76
77     private final NfcAdapter mAdapter;
78     final String mPackageName;
79
80     /** get service handles */
81     private static void initService(NfcAdapter adapter) {
82         try {
83             Method getNfcAdapterExtrasInterface =
84                 NfcAdapter.class.getMethod(
85                     "getNfcAdapterExtrasInterface");
86             final INfcAdapterExtras service = (INfcAdapterExtras)
87                 getNfcAdapterExtrasInterface.invoke(adapter);
88             if (service != null) {
89                 // Leave stale rather than receive a null value.
90                 sService = service;
91             }
92         } catch (Exception e) {}
93     }

104     public static NfcAdapterExtras get(NfcAdapter adapter) {
105         Context context = null;
106         try {
107             Method getContext =
108                 NfcAdapter.class.getMethod("getContext");
109             context = (Context)getContext.invoke(adapter);
110         } catch (Exception e) {}
111         if (context == null) {
112             throw new UnsupportedOperationException(
113                 "You must pass a context to your NfcAdapter to use the
114                     NFC extras APIs");
115         }
116         synchronized (NfcAdapterExtras.class) {
117             if (sService == null) {
118                 initService(adapter);
119             }
120             NfcAdapterExtras extras = sNfcExtras.get(adapter);
121             if (extras == null) {
122                 extras = new NfcAdapterExtras(adapter);

```

```

123     sNfcExtras.put(adapter, extras);
124     }
125     return extras;
126     }
127 }
128
129 private NfcAdapterExtras(NfcAdapter adapter) {
130     mAdapter = adapter;
131     String packageName = null;
132     try {
133         Method getContext =
134             NfcAdapter.class.getMethod("getContext");
135         packageName = ((Context)getContext.invoke(adapter))
136             .getPackageName();
137     } catch (Exception e) {}
138     mPackageName = packageName;
139     mEmbeddedEe = new NfcExecutionEnvironment(this);
140     mRouteOnWhenScreenOn = new CardEmulationRoute(
141         CardEmulationRoute.ROUTE_ON_WHEN_SCREEN_ON,
142         mEmbeddedEe);
143 }

148 public final static class CardEmulationRoute {

153     public static final int ROUTE_OFF = 1;

159     public static final int ROUTE_ON_WHEN_SCREEN_ON = 2;

164     public final int route;

170     public final NfcExecutionEnvironment nfcEe;
171
172     public CardEmulationRoute(int route,
173         NfcExecutionEnvironment nfcEe) {
174         if (route == ROUTE_OFF && nfcEe != null) {
175             throw new IllegalArgumentException(
176                 "must not specify a NFC-EE with ROUTE_OFF");
177         } else if (route != ROUTE_OFF && nfcEe == null) {
178             throw new IllegalArgumentException(
179                 "must specify a NFC-EE for this route");
180         }
181         this.route = route;
182         this.nfcEe = nfcEe;
183     }
184 }

189 void attemptDeadServiceRecovery(Exception e) {
190     Log.e(TAG,
191         "NFC Adapter Extras dead - attempting to recover");
192     try {
193         Method attemptDeadServiceRecovery =
194             NfcAdapter.class.getMethod(
195                 "attemptDeadServiceRecovery",
196                 Exception.class);
197         attemptDeadServiceRecovery.invoke(mAdapter, e);
198     } catch (Exception ee) {}

```





```

285     }
286
287     return s.toString();
288 } else {
289     return "";
290 }
291 }

302 public void authenticate(byte[] token) {
303     try {
304         Log.d(TAG,
305             "authenticate() for " +
306             ((mPackageName != null) ? mPackageName : "[null]") +
307             ": " + convertByteArrayToHexString(token));
308         sService.authenticate(mPackageName, token);
309     } catch (Exception e) {
310         attemptDeadServiceRecovery(e);
311     }
312 }
313 }

```

## B.2 Class NfcExecutionEnvironment

```

1  /*
2  * Copyright (C) 2011 The Android Open Source Project
3  * Modifications (debug output): (C) 2012 Michael Roland
4  *
5  * Licensed under the Apache License, Version 2.0 (the
6  * "License"); you may not use this file except in compliance
7  * with the License. You may obtain a copy of the License at
8  *
9  *     http://www.apache.org/licenses/LICENSE-2.0
10 *
11 * Unless required by applicable law or agreed to in writing,
12 * software distributed under the License is distributed on
13 * an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY
14 * KIND, either express or implied. See the License for the
15 * specific language governing permissions and limitations
16 * under the License.
17 */
18
19 package com.android.nfc_extras;
20
21 import android.os.Binder;
22 import android.os.Bundle;
23 import android.os.RemoteException;
24 import android.util.Log;
25
26 import java.io.IOException;
27
28 public class NfcExecutionEnvironment {
29     private static final String TAG="NfcExecutionEnvironment";
30
31     private final NfcAdapterExtras mExtras;
32     private final Binder mToken;

```

```

46  public static final String ACTION_AID_SELECTED =
47      "com.android.nfc_extras.action.AID_SELECTED";

```

```

55  public static final String EXTRA_AID =
56      "com.android.nfc_extras.extra.AID";

```

```

70  public static final String ACTION_APDU_RECEIVED =
71      "com.android.nfc_extras.action.APDU_RECEIVED";

```

```

80  public static final String EXTRA_APDU_BYTES =
81      "com.android.nfc_extras.extra.APDU_BYTES";

```

```

88  public static final String ACTION_EMV_CARD_REMOVAL =
89      "com.android.nfc_extras.action.EMV_CARD_REMOVAL";

```

```

102 public static final String ACTION_MIFARE_ACCESS_DETECTED =
103     "com.android.nfc_extras.action.MIFARE_ACCESS_DETECTED";

```

```

113 public static final String EXTRA_MIFARE_BLOCK =
114     "com.android.nfc_extras.extra.MIFARE_BLOCK";

```

```

115
116 NfcExecutionEnvironment(NfcAdapterExtras extras) {
117     mExtras = extras;
118     mToken = new Binder();
119 }

```

```

135 public void open() throws IOException {
136     try {
137         Log.d(TAG,
138             "open() for " +
139             ((mExtras.mPackageName != null) ?
140              mExtras.mPackageName : "[null]"));
141         Bundle b = mExtras.getService().open(mExtras.mPackageName,
142                                             mToken);
143         throwBundle(b);
144     } catch (Exception e) {
145         mExtras.attemptDeadServiceRecovery(e);
146         throw new IOException("NFC Service was dead, try again");
147     }
148 }

```

```

158 public void close() throws IOException {
159     try {
160         Log.d(TAG,
161             "close() for " +
162             ((mExtras.mPackageName != null) ?
163              mExtras.mPackageName : "[null]"));
164         throwBundle(mExtras.getService().close(
165             mExtras.mPackageName,
166             mToken));
167     } catch (Exception e) {

```

```

168     mExtras.attemptDeadServiceRecovery(e);
169     throw new IOException("NFC Service was dead");
170 }
171 }
172
173 /**
174  * Convert a byte array into its hexadecimal string form.
175  * @param b Byte array.
176  * @return Hexadecimal string representation.
177  */
178 private static String convertByteArrayToHexString(byte[] b) {
179     if (b != null) {
180         StringBuilder s = new StringBuilder(2 * b.length);
181
182         for (int i = 0; i < b.length; ++i) {
183             final String t = Integer.toHexString(b[i]);
184             final int l = t.length();
185             if (l > 2) {
186                 s.append(t.substring(l - 2));
187             } else {
188                 if (l == 1) {
189                     s.append("0");
190                 }
191                 s.append(t);
192             }
193         }
194
195         return s.toString();
196     } else {
197         return "";
198     }
199 }

```

```

209 public byte[] transceive(byte[] in) throws IOException {
210     Bundle b;
211     try {
212         Log.d(TAG,
213             "transceive() for " +
214             ((mExtras.mPackageName != null) ?
215              mExtras.mPackageName : "[null]") +
216             ": C-APDU=" + convertByteArrayToHexString(in));
217         b = mExtras.getService().transceive(mExtras.mPackageName,
218                                             in);
219     } catch (Exception e) {
220         mExtras.attemptDeadServiceRecovery(e);
221         throw new IOException(
222             "NFC Service was dead, need to re-open");
223     }
224     throwBundle(b);
225     byte[] out = b.getBytes("out");
226     Log.d(TAG,
227         "transceive() for " +
228         ((mExtras.mPackageName != null) ?
229          mExtras.mPackageName : "[null]") +
230         ": R-APDU=" + convertByteArrayToHexString(out));
231     return out;
232 }
233
234 private static void throwBundle(Bundle b) throws IOException {
235     if (b.getInt("e") == -1) {
236         Log.d(TAG, "IOException: " + b.getString("m", "[null]"));
237         throw new IOException(b.getString("m"));
238     }
239 }
240 }

```

# Index

## A

Access control, [62](#), [104](#), [105](#), [108](#), [111](#), [112](#),  
[164](#)  
    bypass, [112](#)  
    enforcer, [110](#)  
Access Rule Applet, [111](#)  
Android, [2](#), [57](#), [107](#), [148](#)  
    vulnerabilities, [58](#)  
Answer-to-Reset, [14](#), [105](#), [110](#)  
Answer-to-Select, [17](#)  
Anti-collision, [16](#)  
APDU, *see* Application Protocol Data Unit  
App, [1](#), [57](#), [120](#), [165](#)  
    signature, *see* Code signing  
Applet, [18](#), [105](#), [142](#)  
Application processor, [55](#), [104](#), [112](#), [120](#),  
[141](#), [165](#)  
Application Protocol Data Unit, [14](#), [105](#),  
[106](#), [108](#), [111](#), [120](#), [131](#)  
ARA, *see* Access Rule Applet  
ARM TrustZone, [59](#)  
ATR, *see* Answer-to-Reset  
ATS, *see* Answer-to-Select  
Authenticity, [79](#)  
Authorization, [79](#), [83](#)  
Automotive computer system, [37](#)

## C

CA, *see* Certification authority  
Car  
    immobilizer, [34](#)  
    key, [34–36](#)  
Card emulation, [2](#), [5](#), [21](#), [27](#), [40](#), [44](#), [62](#), [103](#),  
[164](#), [168](#)  
Card emulator, [121](#), [124](#), [128](#), [155](#), [165](#)  
Card Manager, [18](#), [115](#), [131](#), [149](#)

Certificate, [81](#), [105](#), [109](#), [111](#)  
    content mapping, [81](#)  
    format, [89](#)  
    lifespan, [86](#), [164](#)  
Certification authority, [79](#)  
Chain model, [87](#)  
Chip & PIN, [29](#), [60](#)  
Cloning, [56](#)  
Cloud, [38](#)  
Code signing, [57](#), [85](#), [106](#), [107](#), [112](#)  
Collision resistance, [72](#)  
Command-response delay, [132](#), [137](#), [156](#)  
Common Criteria, [49](#)  
Compute Cryptographic Checksum, [153](#)  
Connection handover, [27](#), [33](#), [36](#), [38](#), [40](#), [42](#),  
[43](#), [82](#)  
Content spoofing, [52](#)  
Credential storage, [103](#)  
Crypto-1 cipher, [48](#)

## D

Data insertion, [47](#)  
Data manipulation, [71](#)  
Data modification, [47](#), [56](#)  
Denial-of-service, [62](#), [97](#), [115](#), [117](#), [142](#), [165](#)  
Digital signature, [53](#), [54](#), [62](#), [72](#), [166](#)  
    partial, [74](#), [82](#), [83](#), [97](#)  
    scope, [74](#), [78](#)  
    trust, [79](#), [91](#), [97](#), [100](#)  
Distance-bounding, [51](#)

## E

Eavesdropping, [43](#), [47](#)  
EMV, [28](#), [50](#), [54](#), [60](#), [123](#)  
    cloud-based, [142](#)

EMV mode, *see* Chip & PIN  
mag-stripe mode, 29, 151

## F

FDT, *see* Frame delay time  
Feature phone, 1  
FeliCa, *see* JIS X 6319-4  
Fleet management, 36  
Frame delay time, 122  
Frame waiting time, 123  
Fuzzing, 53  
FWT, *see* Frame waiting time

## G

Get Processing Options, 152, 153  
GlobalPlatform, 18, 42, 115, 116  
Google Wallet, 5, 60, 108, 147, 148, 165  
  on-card component, *see* On-card component  
  PIN, 154, 157  
  relay attack, 165  
  unlock command, 154, 157

## H

Hash function, 72  
HCE, *see* Host-based card emulation  
Host-based card emulation, 2, 21, 28, 56, 120, 125, 155, 168

## I

In-browser payment, 54, 143  
Inductive coupling, 15  
ISO/IEC 14443, 15–17, 28, 122  
ISO/IEC 15693, 17  
ISO/IEC 18092, 20  
ISO/IEC 7816, 13, 14, 18

## J

Jail breaking, *see* Privilege escalation  
Java Card, 18  
JIS X 6319-4, 17

## L

LLCP, *see* Logical Link Control Protocol  
Logical Link Control Protocol, 20, 48

## M

Mafia fraud, *see* Relay attack

MasterCard PayPass, 29, 60, 147, 151  
MIFARE Classic, 48

## N

NDEF, *see* NFC Data Exchange Format  
Near Field Communication, 1, 19  
  Card emulation mode, *see* Card emulation  
  key, 34–36  
  operating modes, 20  
  Peer-to-peer mode, 20, 40, 42  
  Reader/writer mode, 21, 40, 43, 106  
  Record Type Definition, 24  
  security, 3, 4, 47, 51, 163, 167  
  tag, *see* Tagging  
  wired interface, 28  
NFC, *see* Near Field Communication  
NFC Data Exchange Format, 22, 69  
  API, 77  
  chunk flag, 22, 24, 76  
  connection handover, *see* Connection handover  
  external type, 24  
  message, 24  
  parser, 77  
  record, 22  
  short record, 23, 75  
  signature, 73, *see also* Signature Record Type Definition  
  smart poster record, *see* Smart poster  
  text record, 25, 81  
  type name format, 23, 76, 77, 93  
  URI record, 25, 81  
  well-known type, 24  
NFC Forum, 3, 20, 54, 88, 166  
Normalized form, 77, 78, 99

## O

On-board credentials, 59  
On-card component, 149, 150, 154  
On-device secure payment, 158  
Online signature generation, 85, 164  
Open Mobile API, 109, 110  
Over-the-air management, 42, 55, 104, 115, 141, 142

## P

Pairing  
  out-of-band, *see* Connection handover  
PC/SC, *see* Personal Computer/Smart Card interface

PCD, *see* Proximity Coupling Device  
 Personal Computer/Smart Card interface,  
 19, 131  
 Personalization, 34  
 PICC, *see* Proximity Integrated Circuit Card  
 PKI, *see* Public-key infrastructure  
 Power analysis, 49  
 Preimage resistance, 72  
 Private key, 38, 80, 81, 84, 85  
 Privilege escalation, 53, 57, 60, 114, 120,  
 147, 165  
   framework, 58, 154  
 Proximity Coupling Device, 16  
 Proximity Integrated Circuit Card, 16  
 Public-key infrastructure, 79, 91, 99, 163,  
 167

**R**

Record composition attack, 62, 96, 97, 164  
 Record hiding, 93  
 Record joining, 93, 95  
 Relay attack, 5, 50, 56, 104, 115, 118  
   countermeasures, 51, 141

**S**

Secret key, *see* Private key  
 Secure element, 4, 21, 27, 37, 40, 41, 44, 54,  
 103, 164, 168  
   API, 62, 104, 127, 148  
   attacks, 114  
   cloud-based, 124, 168  
   external mode, 40, 41, 130, 135  
   internal mode, 40, 41, 104, 115, 118, 120,  
   130, 135, 141, 142, 158  
 Secure Element Evaluation Kit, 109  
 Security domain, 18, 105, 115  
 SEEK, *see* Secure Element Evaluation Kit  
 Shell model, 86  
 Signature Record Type Definition, 4, 54, 62,  
 73, 88, 163  
   coverage, 90, 92  
   weaknesses, 90, 98  
 SIM card, *see* Universal Integrated Circuit  
 Card  
 Single Wire Protocol, 28  
 Skimming, 55, 104  
 Smart phone, 1  
   app, *see* App

  security, 57  
 Smart poster, 26, 52, 69, 71, 81, 83, 97  
 Smartcard, 4, 13, 27, 40, 103, 106  
   attack, 49  
   file system, 14  
   lifecycle, 116  
   protocol stack, 14  
   security, 44  
 Soft-SE, *see* Host-based card emulation  
 Software card emulation, *see* Host-based  
 card emulation  
 Software-based relay attack, 62, 115, 118,  
 120, 147, 154, 165  
   prototype, 126  
   solutions, 157  
 Spoofing, 53, 70, 72  
 Subscriber Identity Module, *see* Universal  
 Integrated Circuit Card  
 SWP, *see* Single Wire Protocol

**T**

Tag-length-value format, 19  
 Tagging, 4, 5, 21, 33, 40, 43, 52, 61, 69, 163  
   security, 52, 70  
 TLV, *see* Tag-length-value format  
 Trusted computing, 59, 141, 168  
 Trusted service manager, 42, 104  
 TrustZone, *see* ARM TrustZone  
 Two-factor authentication, 142

**U**

UICC, *see* Universal Integrated Circuit Card  
 Universal Integrated Circuit Card, 13, 27, 42,  
 105, 115  
 Usage data, 92, 164

**W**

Wallet  
   digital, 41, 59  
   Google, *see* Google Wallet  
 Wormhole attack, 50  
 Write protection, 52, 71

**Z**

Zapper, 72