tinyDTLS Experiment Record

Yan Yan

March 2, 2015

Contents

1 Toys **2**
1.1 Odd or Even 2
1.1.1 Description 2
1.1.2 Analysis [To be completed...] 3
1.2 Leaky Coffee 3
1.2.1 Description 3

Chapter 1

Toys

In this hello-world set-up, there is only one server and one client connected through local-link. The protocol suit we adopted is: [IPv4 or IPv6] + UDP + DTLS.
Abbreviations:

CLIENT Client.

SERVER Server.

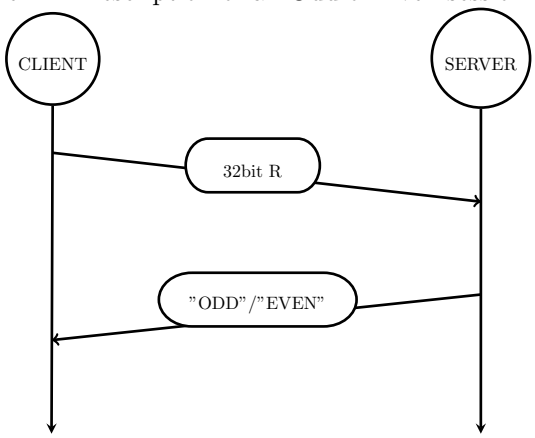
1.1 Odd or Even

Odd or Even is an extremely simple toy application. It is designed to demonstrate the fundamental idea of traffic analysis.

1.1.1 Description

CLIENT randomly generates a 32-bit unsigned integer R and sends it to SERVER. SERVER replies with a string "ODD" or "EVEN" according to the integer sent.

Figure 1.1: Description of an Odd-or-Even session



1.1.2 Analysis [To be completed...]

For every Odd-or-Even session,

Packets from CLIENT to SERVER:

All fields for every packet are the same, except: 1. Encrypted Application Data field in DTLS layer. 2. Sequence Number increased by 1 every packet. 3. Checksum in UDP layer.

Packets from SERVER to CLIENT:

All fields are the same for every packet except: 1. Encrypted Application Data field in DTLS layer. 2. Sequence Number increased by 1 every packet. 3. Checksum in UDP layer. 4. Length field in both DTLS layer and UDP layer. The values are always (20,41) respectively when data is "Odd" and (21,42) when data is "Even".

Therefore in this application, given pre-knowledge that server responds with either "Odd" or "Even", the length field in both DTLS layer and UDP layer can directly leak the plaintext.

1.2 Leaky Coffee

1.2.1 Description

Leaky Coffee simulates the scenario that CLIENT initiates a Leaky-Coffee session with a request to SERVER, SERVER replies with a response and CLIENT then reacts according to the response.

Syntax

Definition 1.2.1. *COFFEE* is a set of strings defined as:
$COFFEE = \{$"AMERICANO","CAPPUCCINO","ESPRESSO","MOCHA"$\}$

Definition 1.2.2. Let '*' represents SUGAR and '@' represents MILK respectively, we denote n_* and $n_@$ as the number of appearances of '*' and '@' in a string. We also call n_* and $n_@$ the degree of SUGAR and MILK of a string.

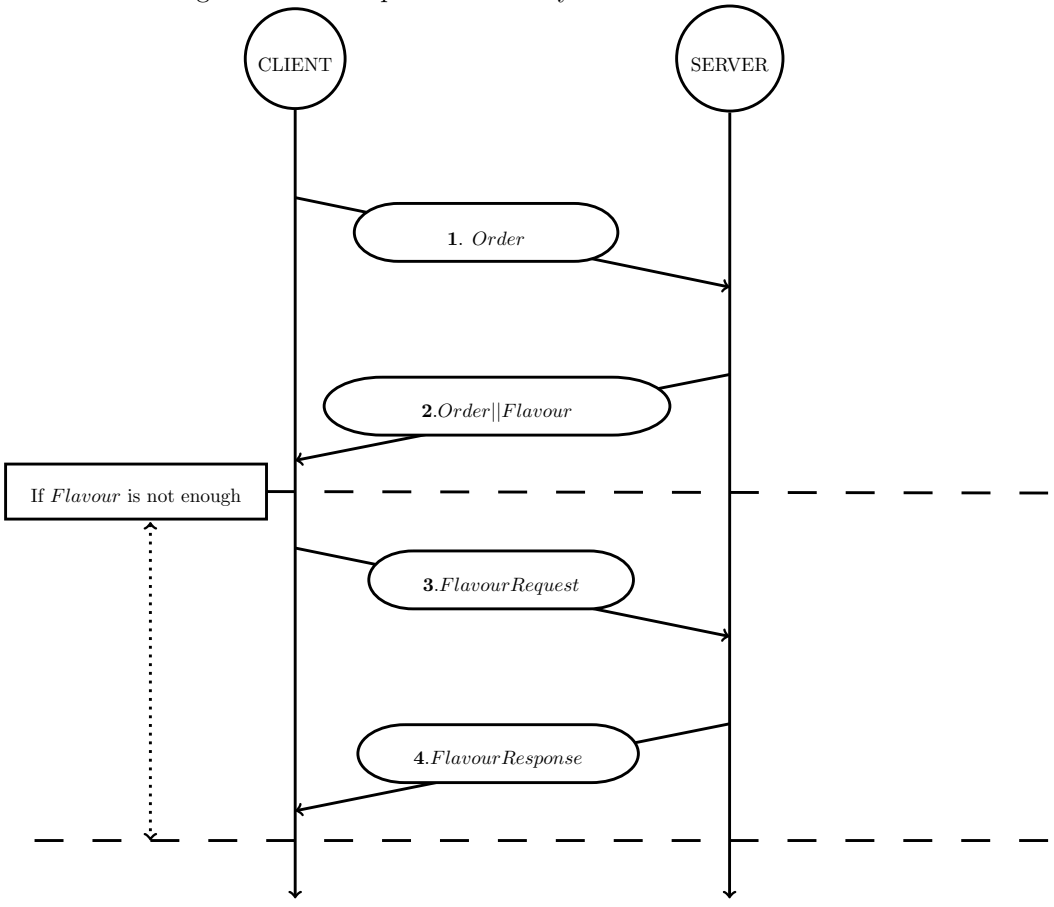
Definition 1.2.3. We define a set of string $ADDITIVE$ as:
$ADDITIVE = \{\{SUGAR, MILK\}^{0-6} | 0 \leq n_* \leq 3, 0 \leq n_@ \leq 3\}$.

In another word, an instance of $ADDITIVE$ contains no more than 3 SUGAR and MILK.

Leaky-Coffee Session

A Leaky-Coffee session can be described as:

Figure 1.2: Description of a Leaky-Coffee session



1 As an initiation of a conversation, CLIENT randomly picks a string $Order \in COFFEE$ and sends it to SERVER.

2 Upon receiving an *Order*, SERVER replies with a string $\{Order||Flavour\}$ where $Flavour \in ADDITIVE$ and $||$ represents concatenation. If $Order =$ "ESPRESSO" then the degrees of both SUGAR and MILK of *Flavour* are set to 0.

3 CLIENT randomly generates a SUGAR requirement $r_* \in [0, 3]$ and a MILK requirement $r_@ \in [0, 3]$. Then it scans the reply from **2** and computes its degrees of SUGAR and MILK. If any of the degrees does not met the requirements, i.e. $n_* < r_*$ and/or $n_@ < r_@$, then CLIENT sends a $FlavourRequest = \{"FLAVOUR"||\{SUGAR\}^{\max(r_*-n_*,0)}||MILK^{\max(r_@-n_@,0)}\}$.

4 If SERVER receives a *FlavourRequest*, it echoes back *FlavourRequest* as its *FlavourResponse*, i.e. $FlvaourResponse = FlavourRequest$.

Note that the *FlavourRequest* and *FlavourResponse* packets are probabilistic in a Leaky-Coffee Session.

Example 1.2.1. An example with *FlavourRequest* and *FlavourResponse*:

1

Bibliography