Verification of Drivers License

2017010- Agnel Aaron
2017035 - Bharath Kumar

The problem we are trying to solve involves police officers (There are quite a few) checking the license of the driver, so the system that we will build needs to authenticate the user (police) and there are multiple users. Also note that since India has the 2nd largest population, we also need to have multiple servers. Kerberos can be used to solve the problem, however we have modified it to better suit our needs.

Setup

We know that kerberos needs at least 1 ticketing server (let's call it kdc) and 1 file server. The ticketing server issues the 2 tickets and we can use the 2nd ticket to get access to a service provided by the file server.

Our system will have multiple ticketing servers and multiple file servers to try and simulate the concept of 'realms'. Each of these file servers will have a dictionary that maps unique license numbers to ['expiry date','photo','2 or 4 wheeler or both']. We will make sure that all licenses will exist on only 1 file server, i.e the pairwise intersection of entries of file servers is the null set.

Modified Unique License Number

We will ensure the above property the following way. Assume there are a total of n file servers in india.If we represent them as a **complete binary tree**, the tree will have h(height) = ceil(log2(n+1))-1. Since no 2 servers are in the same place in the tree we can encode each server with help of the fact that each position in the tree is occupied by at most 1 license. For example if the code is '1001', we traverse from LSB to MSB, and we go right, then left, then left, then right. Note that the length of the code will be h bits and if the i'th bit is 0 then on the h-i+1'th step we go left else go right.

Each license will have to be in 1 file server, so the first h bits of the license will be reserved for the code, and the remaining bits will be generated uniquely.

Keys

We use symmetric keys predominantly, however we also use private-public key pair to sign hashes .
Kc - Symmetric,known only to kdc and client(C)
Kx - Symmetric,known only to kdc
Kv - Symmetric,known only to kdc and a particular file server
K_pub/K_priv - UnSymmetric, private key known only to kdc

Modified Kereberos

1.C → KDC
X = E(Kc,"Hello"||T1)||IDc||E(Kc,hash)
E(K_pub,X)
2. KDC now has authenticated Client
3. KDC → C
E(Kx,IDc||lifetime1) This is Ticket1
Log in phase over
4.C → KDC
E(Kc,Ticket1)||license_number(1..h)||E(Kc,hash)
5.KDC verifies authenticity of ticket and finds out the correct file server based on the first h bits say v is the name of the file server.
6. KDC → C
E(Kc,Ticket2)
Ticket2 = E(Kv,IDc||lifetime2||Kc)
The Kc will be used for further communication b/w client C and file server v
7.C → File Server V
Ticket2
8. V → C
E(kc,['expiry date','photo','2 or 4 wheeler or both'])

<u>Questions</u>

1) Driver Provides his U.M.L (Unique Modified License number) to the police. The police uses our system and our system will return him Photo id - so that the police can first authenticate the driver, expiry date + type of vehicle - so that police can check if his license hasn't expired or he is driving the wrong vehicle type.
2) No need for a central server, as it would perform horribly due to India's driving population (bottleneck). Instead of this we have made a system with multiple file servers (servers with license details) that contain mutually exclusive entries, and we can divide these servers regionally and put licenses belonging to that particular region.
3) Yes, otherwise the system could be exploited by capturing the packet and sending later (replay attacks), We have lifetime in each of our tickets and also include current time (T1) in the first query from C→ KDC. As a result every query has a time/lifetime field therefore our system is immune to replay attacks.
4) Signatures can be used to confirm who sent the message, so in our case, every step is encrypted by a symmetric key that only 1 person other than the party receiving the message knows, and there is lifetime/time, so after confirming its not replay, we know for sure who sent the message. Example Ticket1 is encrypted by Kx which no one other than kdc knows, so the Ticket1 is signed by kdc and is verified by kdc later.
5) Data integrity is preserved by adding an encrypted hash at the end of a message as if someone changes content, he cannot change hash as it is encrypted, and upon receiving we will recalculate hash and verify integrity . We also add encrypted hashes in multiple steps throughout our algorithm.
6) Confidentiality, Integrity and authentication is relevant here and has been preserved. As explained above after confirming it's not a replay, we can detect who sent/signed a message, so non-repudiation is also preserved.