The TESLA Broadcast Authentication Protocol

R07944058
網媒碩一 陳鵬宇

1 Summary

為什麼我們需要認證？為了確保封包的 sender「確實」是該 sender ，並且沒有在中途被篡改。在廣播訊息系統中，我們可以對每一個封包提供一個唯一的安全簽章，但這樣會在簽署和驗證封包時花費大量成本，同時頻寬也不允許。

我們也可以同時對所有封包進行一次性的簽署，但這會有巨大的漏洞，攻擊者可以通過用虛假地發送數據包給接收方來引入 DOS（拒絕服務）攻擊。因為數位簽章很計算成本大，接受者光驗證這些封包的正確性就必需耗費大量時間。

此篇論文提出了一個在廣播訊息認證時的協定 TESLA，他能忍受一定程度的封包丟失。TESLA 基於 sender 與 receiver 之間 loose time synchronization，並且巧妙的運用了 one-way function 的性質，利用時間差和一次性金鑰去確保鑰匙（key）和訊息（M）的安全。

2 Strength(s)

此篇論文提到 receiver (R) 必需要在 $t = t_R$ 時送給 sender (S) 一個 Nonce，並且 S 在收到當下 ($t = t_S$)，馬上回傳剛收到的 Nonce 和 t_S 給 R，目的是計算出時間同步最大時間誤差 Δ。真實的傳送時間誤差為 $\delta = t_S - t_R$，我們不需要這麼精確，透過 full round-trip time (RTT) 我們更能計算出稍微「鬆散」的時間差，同時確保資料傳遞上的安全性。TESLA 的實作方式會在 time interval 為 T_{i+d} 時，公布 K_i，我們知道 key 必需在 Δ 時間差內被 R 收到，否則無法知道這把 key 是否安全，也無法使用此 key 去計算 MAC。

One-way chain 是反向產生的，所以任何 K_i，都能透過 $K_j (j > i)$，經過函數 $F^{j-i}(K_j)$ 產生 K_i，因此可以能夠承受一定承度的封包丟失，同時也能用舊有的 K_i 來驗證新收到的 $K_j (j > i)$ 是否正確。

3 Weakness(s)

時間同步需要在雙方開始溝通前，receiver 首先送出 Nonce 給 sender 來計算出 Δ，這是時間同步協定的核心，但如果攻擊者可以猜測到 receiver 的 Nonce，他就可以偽裝自己是 receiver 利用該 Nonce 送一個時間同步的請求給 sender，然後攻擊者再對 receiver 重複做一樣的動作，去混淆 sender 讓 sender 誤以為 receiver 已經收到他傳的 Nonce 了，因此我們必需要確保 Nonce 在寄送時的渠道是安全的，可以試著在這裡使用一次性的數位簽章增加安全性，如果一開始的這層被攻破，後續的 TESLA protocal 都沒得談了。

4 Reflection

此篇論文很完整的將 TESLA protocal 的運作方式逐步驟講清楚，從 Sender Setup, Bootstrapping Receivers, Broadcasting Authenticated Messages 到 Authentication at Receiver。計算 Δ 的方式讓我想到有點像運用在 OS 或網路上的 handshaking，當兩個通信裝置或系統之間，需要交換資訊時（就像這裡的 sender 和 receiver），必需事先確定控制信號或字元序列的操作過程，可以用於建立聯接、拆斷聯接（可能是因為不安全）或交換資料等。同時在電腦通信安全中，透過 handshaking 也能驗明單方或雙方身份資料的交換操作過程。

在此篇論文發現一個很有趣的性質，雖然 TESLA 使用的是對稱式的加密方式，但由於 key 要在 d 個時間點後才會由 sender 端揭漏，所以 TESLA 有著非對稱性的性質。

TESLA 巧妙的運用到時間差的方式來作一定時間內的安全驗證。而基於時間上的一次性金鑰，和 Time-based One-Time Password 演算法有點像，Timestamp 通常每 30 秒會增加一次，也被廣泛運用在 Google, Facebook 和 Microsoft 等大公司。