共通文字列を用いた簡単な認証

クレモナのエルマンノ †

E-mail: †privacy@shuudow.dip.jp

あらまし ナップサック問題に基づく、相手認証方式を提案する。この認証法はバイト列を結合して文字列にするのは容易だが、元に戻すことは難しいという一方向性に基づいている。クライアントとサーバはバイト列を事前共有しているものとする。クライアントはサーバから送られてくる乱数と置換ベクトルを用いてバイト列を変換し、更にこのバイト列に join 関数などを適用し文字列として結合させたデータを受信者に送る。オリジナルのバイト列を共有している当事者間では、変換後の文字列が常に一致していることを検証することで相手を認証できる。

キーワード 認証，ナップサック問題，事前共有鍵。

A simple authentication by common strings

Cremona de El manna †

E-mail: †privacy@shuudow.dip.jp

Abstract We propose a authentication method that is easy to implement and with one-way. The one-way of consolidated bytes is based on the fact that it is easy to concatenate bytes but it is difficult to separate to original bytes if there are no knowledge of original bytes' digits and the order. The client and server are assumed to pre-share a sequence of bytes. Using a random number and permutation vector that is sent by the server, the client converts the bytes to other bytes, and sends it as a string applied some join like function to saver. The parties that shared the original bytes can authenticate the other party by verifying that the converted string is always consistent to result of server.

Keyword authentication, knapsack problem, pre-shared key

1. 認証

1.1. 本稿で述べる認証について

認証とはデータ及び端末を、システムが許可している権限を持っているかどうかを識別する方法として使われている。一般には、元に戻すことのできないハッシュ関数を使って認証している。ここでは複雑なハッシュ関数に替わって、ランダムに生成した置換群で変換したバイト列を認証に用いることにより、ワンタイムパスワードを構成することを目的とする。この方法のメリットには、次のものが挙げられる。

第三者から見ると乱数をやりとりしているようにしか見えない。
処理が軽く実装が楽である。

共有文字列を用いた認証の他の方式については、参考文献 2 がある。

1.2. パラメータ

秘密鍵：ランダム行列かランダム線形符号の部分符号 A、パスワード x（32 バイト）。x を行列 A で変換したバイト列 s。

これらを、クライアントとサーバで共有する。

2. バイト列を用いたナップサック問題の構成

バイト列を文字列として連結させると、各要素の桁情報がなくなる為に連結後の文字列を元のバイト列に戻すことが難しくなる。この一方向性は桁ベクトルをナップサックベクトルとしてみなすことでナップサック問題であることがわかる。ナップサック問題を用いてワンタイムパスワード認証を構成する。以下に、その手順を述べる。

・文字列結合による相手認証プロトコル

Step1. 事前にバイト列 s を共有する。

Step2. サーバがクライアントに、ランダムな 0<c<256 と置換群 π を送信する。

Step3. s*c%10000 を計算し、置換した後、バイト列を関数 join で結合した文字列 r を、サーバに送信する。

Step4. サーバはクライアントからの文字列 r と、サーバがクライアントに対して行ったチャレンジを用い、s に対する演算結果が r と一致することを確認する。

サーバは、受信した文字列とクライアントと共有しているバイト列に自分のチャレンジを用いて変換した文字列とを比較し、一致していれば受理する。

3. 今後の課題

バイト列結合の一方向性は、ベクトルの桁の情報が結合後に失われることに基づいている。バイト列の長さと桁を決めることで、この問題は組合せ問題に帰着できる。置換群を直接送信する代わりに時刻同期によって置換群を同時に生成すれば、さらに問題が難しくなる。というのも共有バイト列の長さが攻撃者に分からなくなるからである。共通入力を用いれば、検査するバイト列の長さを可変にすることによって攻撃に対して強くすることができる。また、桁に関する情報を何らかの方法で元に戻す事ができれば、対称、非対称鍵暗号の構成に繋がるものと思われる。

参　考　文　献

[1] Oliver Prezel, Finite fields and Codes, Oxford University Press, London, 1992.

[2] Oblivious Verification of Common String、Claude Cr´epeau and Louis Salvail
http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.17.9227