RAPPORT TECHNIQUE PROUVÉ

A survey of algebraic properties used in cryptographic protocols

Auteur	:	Véronique Cortier, Stéphanie Delaune et Pascal Lafourcade
Date	:	June 30, 2004
Rapport PROUVÉ numéro	:	2
Version	:	1.1

Loria
CNRS UMR 7503,
Campus Scientifique - BP 239
54506 Vandoeuvre-lès-nancy cedex
www.loria.fr

Laboratoire Spécification Vérification
CNRS UMR 8643, ENS Cachan
61, avenue du président-Wilson
94235 Cachan Cedex, France
www.lsv.ens-cachan.fr

Laboratoire Verimag
CNRS UMR 5104,
Univ. Joseph Fourier, INPG
2 av. de Vignate,
38610 Gières, France
www-verimag.imag.fr

Cril Technology
9/11 rue Jeanne Braconnier
92360 Meudon La Foret Cedex, France
www.cril.fr

France Telecom
Div. Recherche et Développement
38, 40 rue du Général Leclerc
92794 Issy Moulineaux Cedex
www.rd.francetelecom.fr

Résumé : Les primitives cryptographiques sont le plus souvent représentées par des symboles libres de fonctions, suivant ainsi l'*hypothèse du chiffrement parfait*. Cependant certaines attaques ou même le déroulement honnête de certains protocoles utilisent les propriétés algébriques d'opérateurs comme le « ou exclusif », l'exponentiation modulaire ou l'addition par exemple.
Nous faisons ici un état de l'art des propriétés algébriques pertinentes pour les protocoles cryptographiques.

A survey of algebraic properties used in cryptographic protocols

Véronique Cortier, Stéphanie Delaune et
Pascal Lafourcade

June 30, 2004

Abstract: Using the *perfect encryption assumption*, cryptographic primitives are often represented by free function symbols. However some attacks and even honest runs may use algebraic properties of the operators like the exclusive or, the modular exponentiation, the addition, etc.
We give here a survey of protocols and attacks using such algebraic properties.

The verification of cryptographic protocols deserved a lot of attention in the past few years, because of the huge application domain of secure communication via public channels. In order to obtain the automation of the verification of these protocols, a common hypothesis is the *perfect encryption assumption*, which ensures that an attacker cannot get any information from an encrypted message except if he has the key. This leads to the representation of messages by terms where e.g. the encryption of a message m by a key k is represented by the term $\{m\}_k$ where $\{\}$ is a free function symbol.

Lots of decidability results have been obtained under this perfect encryption hypothesis: the secrecy preservation is co-NP-complete for a bounded number of sessions [RT01], and decidable for an unbounded number of sessions, under some restrictions [Low98, DLMS99, CLC03]. Lots of tools like [Low97, Bla01, JRV00] have also been developed to automatically verify cryptographic protocols.

However, the primitives used in cryptographic protocols do have algebraic properties. For example, the exclusive or operation is associative, commutative and nilpotent. These properties may be used by an intruder to attack a protocol. That is why recent works allow to handle some properties. For example, Y. Chevalier *et al* [CKRT03] showed that the secrecy preservation remains co-NP-complete for bounded protocols with exclusive or. H. Comon-Lundh and V. Cortier [CLC03] showed that the secrecy preservation for protocols with exclusive or is also decidable for an unbounded number of sessions, provided that participants copy at most one unbounded message at each transition.

The aim of the paper is to present a survey of relevant algebraic properties actually used in protocols. This is the first step to determine which properties should be handled in our theoretical results and integrated in our tools. These properties may be either used in the protocol description itself: an honest run may rely of the commutativity of some operator for example, or they may be used for attacking the protocol: for example, the Needham-Schroeder-Lowe protocol is flawed when the encryption has some prefix properties, which happens when the encryption is implemented using the CBC algorithm for example.

The properties are described informally, using equations while the protocols are described using the spore format [spo], which gives a protocol description close to the Clark&Jacob syntax [CJ97].

Contents

1 Commutative encryption

Shamir Three Pass protocol

Author(s): R. Shamir, *unpublished*

Summary: The following protocol [CJ97] enables to exchange a secret message without sharing any initial secret.

Protocol specification

```
A, B :    principal
Ka, Kb :  key
   1.    A  ->  B  :    {M}Ka
   2.    B  ->  A  :    {{M}Ka}Kb
   3.    A  ->  B  :    {M}Kb
```

Requirements

This protocol assumes that encryption is commutative, *i.e.*
`{{x}y}z = {{x}z}y`.

Description of the protocol rules

The agent `A` encrypts his message `M` by `Ka`, then `B` encrypts the message he received by `Kb`. Since `{{M}Ka}Kb={{M}Kb}Ka`, the agent `A` can decrypt it and send `{M}Kb` to `B`. Then, using his key `Kb`, `B` can retrieve `M`.

Attacks

This protocol is subject to a variety of attack [CJ97]. The one we present here uses that the participants are not authenticated.

```
   1.      A    ->  B  :    {M}Ka
   2.    I(B)   ->  A  :    {{M}Ka}Ki
   3.      A    ->  B  :    {M}Ki
```

Now, the intruder can compute the message `M`.

2 Xor

The $\oplus$ symbol denotes the binary operation called xor. The properties of xor are:

- $x \oplus (y \oplus z) = (x \oplus y) \oplus z$ (associativity),
- $x \oplus y = y \oplus x$ (commutativity),
- there is an element 0 such that $x \oplus 0 = 0 \oplus x = x$, and

- $x \oplus x = 0$ (inverse),

This operation is used in many protocols like the John Bull protocol of APM, the standard 802.11 protocol or the TMN protocol, presented in section 4.

John Bull Protocol of APM using Xor

Author(s): P.Y.A. Ryan, S.A. Schneider 1998

Summary: The protocol proposed by J. Bull in [Bul97] establishes a chain of session keys between a fixed number of participants and a server. The following description of the protocol uses just the three principals Alice, Bob and Charlie.

Protocol specification

```
A, B, C, S :  principal
Ka, Kb, Kc :  key
hash{K}{X} :   key, message -> hash
Sign{K}{X} :   key, message -> message, hash
PK, SK :      principal -> key (keypair)
   1.    A  ->  B  :     SignKa{A,B,Na,-} = Xa
   2.    B  ->  C  :     SignKb{B,C,Nb,-} = Xb
   3.    C  ->  S  :     SignKc{C,S,Nc,-}
   4.    S  ->  C  :     A,B,Kab ⊕ hashKa{Na},{A,B,Na}Kab,
                         B,A,Kab ⊕ hashKb{Nb},{B,A,Nb}Kab,
                         B,C,Kbc ⊕ hashKb{Nb},{B,C,Nb}Kbc,
                         C,B,Kbc ⊕ hashKc{Nc},{C,B,Nc}Kbc
   5.    C  ->  B  :     A,B,Kab ⊕ hashKa{Na},{A,B,Na}Kab,
                         B,A,Kab ⊕ hashKb{Nb},{B,A,Nb}Kab,
                         B,C,Kbc ⊕ hashKb{Nb},{B,C,Nb}Kbc
   6.    B  ->  A  :     A,B,Kab ⊕ hashKa{Na},{A,B,Na}Kab
```

Description of the protocol rules

Each principal takes its inputs, appends a nonce and some address information and signs the result using a hash keyed with its private key. He sends the result to the next node. The last node is the server. He can verify all the received and create some new session keys which it conceals by exclusive-oring them with hashes that only the intended recipients can compute. Theses new messages are sent by the same way to all the principals.

Attacks

The attack described in [RS98] is called *domino attack*. The compromise of any key leads to compromise of all other keys. The cause of this attack is the fact that the server sends back a message like:

$$\texttt{Kab} \oplus \texttt{hashKa\{Na\},Kab} \oplus \texttt{hashKb\{Nb\}}$$
$$\texttt{Kbc} \oplus \texttt{hashKb\{Nb\},Kbc} \oplus \texttt{hashKc\{Nc\}}$$

Now, anyone can compute :

$$\texttt{Kab} \oplus \texttt{hashKb\{Nb\}} \oplus \texttt{Kbc} \oplus \texttt{hashKb\{Nb\} = Kab} \oplus \texttt{Kbc}$$

So if only one principal knows a key, he knows all the keys "downstream" from it. This is why this attack is called the *domino attack*.

Remark

This protocol assumes that all the principals are honest.

Standard 802.11 in WEP (Wired Equivalent Privacy) protocol

Author(s): N. Borisov, I. Goldberg, D. Wagner 2001

Summary: The Wired Equivalent Privacy protocol is used in 802.11 networks to protect link-level data during wireless transmission. It is described in detail in the 802.11 standard [L.M99].

Protocol specification

```
A, B:  principal
K:     key
RC4:   initial vector, key -> message
C:     message -> integrity
   1.   A  ->  B  :    v, ( [M,C(M)] ⊕ RC4(v,K) )
```

Description of the protocol rules

The two principals share a key `K`. `A` wants to communicate the message `M` to `B`. The agent `A` takes an initial vector `v` and computes `RC4(v,K)`. The RC4 algorithm generates a *keystream* (*i.e.* a long sequence of pseudorandom bytes) as a function of `v` and `K`. The agent `A` applies the xor operator to `RC4(v,K)` and `P = [M,C(M)]`, where `C(M)` is the *integrity checksum* of the message `M`. Then `B` can decrypt the message because he knows `K` and has received `v`.

Requirements

The checksum function C has the following homomorphic property on the $\oplus$ symbol : $C(X \oplus Y) = C(X) \oplus C(Y)$

Attacks

The attacks described in [BGW01] are based on the property of the xor operator for the first one and on the homomorphic property of the function C for the second one.

The first attack uses the fact that if the encryption of the plain text `P1` is `C1=P1` $\oplus$ `RC4(v,K)` and the encryption of the plain text `P2` is `C2=P2` $\oplus$ `RC4(v,K)` with the same initial vector `v`. Then `C1` $\oplus$ `C2 = P1` $\oplus$ `P2`. So if an intruder knows a plain text and his cipher he can decrypt any message using the same key `K`.

The second attack consists in modifying a message without the receiver noticing it.

Let `M' = M` $\oplus$ `D` and `C` an intercepted cipher. The attacker computes:

`C` $\oplus$ `[D,C(D)] = RC4(v,K)` $\oplus$ `[M,C(M)]` $\oplus$ `[D,C(D)]`
`= RC4(v,K)` $\oplus$ `[M` $\oplus$ `D,C(M)` $\oplus$ `C(D)]`
`= RC4(v,K)` $\oplus$ `[M` $\oplus$ `D,C(M` $\oplus$ `D)]`
`= RC4(v,K)` $\oplus$ `[M',C(M')]`
`= C'`

It is possible to add any pair composed by `D` and `C(D)` to a cipher without the receiver noticing this change.

Remark

The function checksum is implemented by the CRC-32, and this function has the homomorphism property.

3 Abelian Group

Let the + symbol denote the additive binary operation of the Abelian Group (G,+). The properties of the Abelian Group are:

- For all x, y, z $\in$ G, x + (y + z) = (x + y) + z (associativity)
- There is an element 0 $\in$ G such that x + 0 = 0 + x for all x $\in$ G.
- For all x, $\in$ G there exists y such that x + y = 0 (inverse)
- For all x, y $\in$ G, x + y = y + x (commutativity)

We present a protocol for computing an average, using these properties. Lots of protocols use the structure of the Abelian Groups in combination with other properties. One of them will be presented section 6.

Salary Average

Author(s): B. Schneier 1996

Summary: A group of people wants to compute the average of their salaries without anyone declaring his own salary to the others. It is necessary to suppose that each principal has a public key. This protocol is described by [Sch96]. The protocol is given by example for principals Alice, Bob, Charlie and Dorothy.

Protocol specification

```
A, B, C, D :      principal
Ka, Kb, Kc, Kd :  key
PK, SK :          principal -> key (keypair)
   1.    A  ->    B    :     A, {Na + Sa}PK(B)
   2.    B  ->    C    :     B, {Na + Sa + Sb}PK(C)
   3.    C  ->    D    :     C, {Na + Sa + Sb + Sc}PK(D)
   4.    D  ->    A    :     D, {Na + Sa + Sb + Sc + Sd}PK(A)
   5.    A  ->  B,C,D  :     ( Sa + Sb + Sc + Sd ) / 4
```

Attacks

The protocol is flawed because there is no authentication. The following attack has been suggested by Liana Bozga (private communication):

```
   1.      A    ->    B    :     A, {Na + Sa}PK(B)
   4.    B(D)   ->    A    :     D, {Na + Sa + Sb + Sb + Sb}PK(A)
   5.      A    ->  B,C,D  :     ( Sa + Sb + Sb + Sb ) / 4
```

The agent `B` may answer directly to `A`, impersonating `D`. Then `A` answers, thinking that everybody has added his salary. Using the answer of `A`, the agent `B` is able to learn the `A`'s salary.

4 Homomorphism

In this section, we consider operators that verify equalities of the form $f(g(x,y)) = g(f(x), f(y))$.

ECB and corrected Protocol Needham-Schroeder-Lowe

Author(s): G. Lowe, R. Needham et M. Schroeder 1996

Summary:

- Electronic CodeBook (ECB) mode is the most obvious way to use a block cipher. A block cipher encrypts plain text in fixed-sized-n-bits blocks (often n=64). For messages exceeding n bits, the simplest approach is ECB. ECB consists in partitioning the message into n-bits blocks and encrypting each of them separately.

- The famous protocol Needham-Schroeder corrected by G. Lowe [Low95], is used for mutual authentication of the two principals.

Protocol specification

```
A, B :    principal
Na, Nb :  nonce
PK, SK :  principal -> key (keypair)
   1.    A  ->  B   :     {A, Na}PK(B)
   2.    B  ->  A   :     {B, Na, Nb}PK(A)
   3.    A  ->  B   :     {Nb}PK(B)
```

Description of the protocol rules

A sends to B his name and a new nonce encrypted by the public key of B. B confirms to A with the message encrypted by A's public key composed with B's identity, the nonce received and his new nonce. A sends back the nonce of B encrypted by B's public key. Each principal is sure to talk with the right principal. They share the secrets Na and Nb.

Requirements

A basic property of ECB is that `{ABC}k={A}k{B}k{C}k` where the size of the blocks A, B, C is the maximal size accepted by the cryptographic algorithm.

Attacks

An intruder can attack the protocol by playing two sessions of the protocol and extracting `{Na, Nb}PK(A)` from `{B, Na, Nb}PK(A)`. He reuses it to compute `{I, Na, Nb}PK(A)` and trick A to decrypt Nb for him.

```
    i.1.       A    ->    I    :     {A, Na}PK(I)
    ii.1.    I(A)   ->    B    :     {A, Na}PK(B)
    ii.2.      B    ->  I(A)   :     {B, Na, Nb}PK(A)
                              The intruder extracts {Na, Nb}PK(A)
    i.2.       I    ->    A    :     {I, Na, Nb}PK(A)
    i.3.       A    ->    I    :     {Nb}PK(I)
    ii.3.      I    ->    B    :     {Nb}PK(B)
```

Remark

This attack is possible only if the ECB mode is used, and if the size of every nonce and principal name is a multiple of the maximal size of the cipher imposed by the encryption function.

TMN

Author(s): M. Tatebayashi, N. Matsuzaki, and D.B. Newman 1989

Summary: Distribution of a fresh symmetric key and authentication with symmetric keys, trusted server and public keys (only the public key of the server is used) [TMN89] and see also [LR97].

Protocol specification

```
A, B, S :  principal
Ka, Kb :   key
PK, SK :   principal -> key (keypair)
   1.    A  ->  S  :     B, {Ka}PK(S)
   2.    S  ->  B  :     A
   3.    B  ->  S  :     A, {Kb}PK(S)
   4.    S  ->  A  :     B, Kb ⊕ Ka
```

Requirements

The $\oplus$ symbol denotes the exclusive or operation, defined in section 2.

For each session, the server verifies that the keys Ka and Kb have not been used in previous sessions.

The property needed to perform an attack is a homomorphism relation between $*$ (symbol of an Abelian group) and the cryptographic algorithm, more precisely {x}z∗{y}z = {x*y}z, where x and y range over messages and z ranges over public keys.

Attacks

Simmons [Sim94] presents an attack on this protocol. It is based on the fact that two intruders C and D listen the message {Kb}PK(S) exchanged between A and B at step 3 of the protocol. They share their private keys. C and D can replay the protocol with the server.

```
C, D, S :  principal
Kc, Kd :   key
PK, SK :   principal -> key (keypair)
1.    C  ->  S  :     D, {Kc}PK(S)*{Kb}PK(S)  (= {Kc*Kb}PK(S))
2.    S  ->  D  :     C
3.    D  ->  S  :     C, {Kd}PK(S)
4.    S  ->  C  :     D, Kd ⊕ (Kc*Kb)
```

The intruder C sends in the first step {Kc}PK(S)∗{Kb}PK(S) which is equal to {Kc*Kb}PK(S) by the homomorphism property of the cryptographic algorithm. The server takes it like a fresh nonce and plays the rest of the protocol, *i.e.* the server decrypts {Kc*Kb}PK(S) and sends Kd ⊕ (Kc∗Kb) as the last message for

C. By xoring this last message with the key Kd of D, the intruder first deduces (Kc*Kb). Then, since he knows Kc, he can recover Kb using the property of $*$.

Remark

The attack relies on the homomorphic property and not really on the property of the exclusive or. Indeed, consider now the original protocol, replacing the last message Kd $\oplus$ Kc by {Kd}Kc.

```
1.  C  ->  S  :   D, {Kc}PK(S)
2.  S  ->  D  :   C
3.  D  ->  S  :   C, {Kd}PK(S)*{Kb}PK(S)  (= {Kd*Kb}PK(S))
4.  S  ->  C  :   D, {Kd*Kb}Kc
```

There is also an attack, if in the third step, D sends {Kd}PK(S)$*${Kb}PK(S) (= {Kd*kb}PK(S) by the homomorphism property of the cryptographic algorithm). Then the server answers by {Kd*Kb}Kc as the last message for C. C can decrypt it and deduce Kb using the key Kd of D.

Election

Author(s): R. Cramer R. Gennaro B. Schoenmalers 1997

Summary:

This presentation is a slight simplification of [CGS97] and used in [MGA03]. The goal of the protocol is to organize an election with Ai voters and a server S. The server S centralizes the ballots and computes the result of the election.

Protocol specification

```
Ai, S :  principal
Ks :     key
ri :     number
  i.   Ai  ->  S  :   {[Vi]}Ks
```

Description of the protocol rules

Assume that all principals have a public and a private key, and that there are only two voting options (-1 and 1). Let Ai be a registered voter and Vi his vote. Each Vi sends its vote encrypted by the server public's key. Finally the server S computes the result of the election.

Requirements

The principal property of the cryptographic algorithm is :

{m1}K * {m2}K = {m1 + m2}K

This property is verified by cryptographic algorithms such as Elgamal[ElG85], Paillier[Pai99, FPS00].

Remark

The server computes the result with only one decryption. The server computes the product of all messages received and decrypts it. He doesn't know the vote of any single voter. If the result is positive the winner is 1 otherwise the winner is -1.

5 Prefix property

The prefix property is the ability for an intruder to get any prefix of an encrypted message: from a message {x,y}z, he can deduce the message {x}z. This property strongly depends on the encryption algorithm. For example, the ECB algorithm, presented in section 4 has this property. But the ECB algorithm is not used a lot. A relatively good method of encrypting several blocks of data is Cipher Block Chaining (CBC). In such system, the encryption of message block sequence $P_1P_2\cdots P_n$ with the key K is $C_0C_1C_2\cdots C_n$ where $C_0 = I$ (initialization block) and $C_i = \{C_{i-1} \oplus P_i\}_K$. The CBC encryption system has the following property: if $C_0C_1C_2\cdots C_iC_{i+1}\cdots C_n = \{P_1P_2\cdots P_iP_{i+1}\cdots P_n\}_K$ then $C_0C_1C_2\cdots C_i = \{P_1P_2\cdots P_i\}_K$, which we call the prefix property.

We show how this property can be exploited to find a flaw in the Denning Sacco symmetric key protocol and the Needham Schroeder symmetric key protocol.

Denning Sacco Symmetric Key Protocol with CBC

Author(s): D. E. Denning and G. M. Sacco 1981

Summary: Modified version [DS81] of the Needham Schroeder Symmetric Key with timestamps to fix the freshness flaw. Distribution of a shared symmetric key by a trusted server and mutual authentication, using symmetric key cryptography and timestamps.

Protocol specification

```
A, B, S :        principal
Kas, Kbs, Kab :  key
T :              timestamp
   1.    A  ->  S  :    A, B
   2.    S  ->  A  :    {B, Kab, T, {A, Kab, T}Kbs}Kas
   3.    A  ->  B  :    {A, Kab, T}Kbs
```

Description of the protocol rules

The shared symmetric key established by the protocol is `Kab`.

Attacks

Y. Chevalier and L. Vigneron [CV02] present an attack on this protocol. It is based on the fact that messages 2 and 3 are of the same global shape. It requires that the length of agents names, nonces and timestamps are a multiple of the block length used for encryption.

```
1.   I(B)  ->    S    :   B, A
2.     S   ->  I(B)   :   {A, Kab, T, {B, Kab, T}Kas}Kbs
3.   I(A)  ->    B    :   {A, Kab, T}Kbs
```

Using the answer `{A, Kab, T, {B, Kab, T}Kas}Kbs` of the server and the prefix property, the intruder is able to get the message `{A, Kab, T}Kbs`. Receiving this message, `B` accepts a new value for the symmetric key he shares with `A`, whereas `A` is not aware that a protocol run took place.

Needham Schroeder Symmetric Key with CBC

Author(s): Roger Needham and Michael Schroeder 1978

Summary: Distribution of a shared symmetric key[NS78] by a trusted server and mutual authentication.

Protocol specification

```
A, B, S:         principal
Na, Nb:          nonce
Kas, Kbs, Kab:   key
dec:             nonce -> nonce
1.   A  ->  S  :   A, B, Na
2.   S  ->  A  :   {Na, B, Kab, {Kab, A}Kbs}Kas
3.   A  ->  B  :   {Kab,A}Kbs
4.   B  ->  A  :   {Nb}Kab
5.   A  ->  B  :   {dec(Nb)}Kab
```

Description of the protocol rules

This protocol establishes the fresh shared symmetric key `Kab`.

Messages `1-3` perform the distribution of the fresh shared symmetric key `Kab` and messages `4-5` are for mutual authentication of `A` and `B`.

The operator `dec` is decrementation.

Requirements

The protocol must guaranty the secrecy of `Kab`: in every session, the value of `Kab` must be known only by the participants playing the roles of `A`, `B` and `S` in that session.

If the participant playing `B` accepts the last message `5`, then `Kab` has been sent in message `3` by `A` (whose identity is included in the cipher of message `3`).

Attacks

Beyond other existing attacks, O. Pereira and J.-J. Quisquater [PQ00] presented the following flaw, based on the prefix property.

```
i.1.     A     ->   S     :  A, B, Na
i.2.     S     ->   A     :  {Na, B, Kab, {Kab, A}Kbs }Kas
ii.3.    I(B)  ->   A     :  {Na, B}Kas
ii.4.    A     ->   I(B)  :  {Naa}Na
ii.5.    I(B)  ->   A     :  {dec(Naa)}Na
```

Suppose that message the message `{Na, B, Kab, {Kab, A}Kbs}Kas` has ciphertext $C_0C_1C_2\cdots C_n$ and that all components have length one block. Then `{Na, B}Kas` has ciphertext $C_0C_1C_2$. This message is of the form of an expected message by `A` at the third step of a session initialized by `B`. Thus `A` can be fooled into accepting the publicly known nonce `Na` as a secret key shared with `B`.

6 Abelian group and modular exponentiation

Contactless Electronic Purse

Author(s): M. Girault, J.C. Paillès 2001

Summary: Transaction between an Electronic Purse card (EP) and a Secure Application Module (SAM) based on discrete log approach for fast debit transaction [GP01].

Protocol specification

`EP, SAM, TTC :`	`principal`
`Np, Ns, c, M, s :`	`number`
`Facqkey, Fisskey :`	`principal -> key`
`PK,SK :`	`principal -> key (keypair)`
`h :`	`number -> number`
`Cert :`	`const`

alias `p = exp(b,-s)`

alias `S6 = h(Facqkey(EP), SAM, Ns, Np, M)`

```
 1.    EP   ->  SAM   :    EP, Np, p, Cert
 2.    SAM  ->   EP   :    SAM, Ns, h(Facqkey(EP),EP, Np)
 3.         ->   EP   :    M
EP chooses one coupon (x, exp(b, x))
 4.    EP   ->  SAM   :    h(exp(b,x), SAM, Ns, Np, S6, M)
 5.    SAM  ->   EP   :    c
bal_EP = bal_EP - M
 6.    EP   ->  SAM   :    x+s × c, S6, M
SAM computes u = exp(p,c) × exp(b,y)
 verifies that msg 4 = h(u, SAM, Ns, Np, S6, M)
 bal_SAM = bal_SAM + M
 stores S6
```

Description of the protocol rules

`Facqkey` and `Fisskey` are master keys whose values are initially known only by SAM and TTC respectively. The Electronic Purse (`EP`) only knows the values `Facqkey(EP)` and `Fisskey(EP)`.

The Electronic Purse `EP` sends to the Secure Application Module `SAM` its identity, a challenge, its public-key `p` and a certificate `Cert` (`msg 1`). `SAM` replies to `EP`'s challenge and gives also its identity and a new challenge (`msg 2`). Then `EP` receives the amount of the transaction (`msg 3`). Afterwards, `EP` builds a hash message which contains `exp(b,x)` which depends on a fresh value `x`. The problem of the computation of the value `exp(b,x)` in `EP` is solved by pre-computation, *i.e.* using the "coupon" technique. Coupons are stored in `EP` and they may be computed when reloading the purse. Then, `msg 4` is sent to `SAM` who stores it and generates a fresh number `c` which is sent on the network. At this moment, `EP` debits its balance and sends to `SAM` a response (`msg 6`) which allows him to verify the hash message previously sent thanks to algebraic properties of operators `+`, $\times$ and `exp`. If the verification is correct, then `SAM` credits its balance. Moreover, he stores the ciphertext `S6` for further control.

Requirements

The $+$ and $\times$ symbols denote respectively the addition and multiplication, and exp the modular exponentiation. The axioms listed below are the minimal ones required to perform the verification test at the last step of the protocol.

- $x + y = y + x$ (E1)
- $x + (y + z) = (x + y) + z$ (E2)
- $(-x) + x = 0$ (E3)
- $x + 0 = x$ (E4)

- exp(exp(x, y), z) = exp (x, y × z)   (E5)
- exp(x, y) × exp(x, z) = exp(x,y+z)   (E6)

Details of the last calculus performed by `SAM`:

```
exp(p,c) × exp(b,y) = exp(exp(b, -s),c) × exp(b,x + s × c)
                    = exp(b, -s × c) × exp(b,x + s × c)   (E5)
                    = exp(b, -s × c + (x + s × c))        (E6)
                    = exp(b, (-s × c + s × c) + x)        (E1,E2)
                    = exp(b, 0 + x)                       (E3)
                    = exp(b, x)                           (E4)
```

Since the modular exponentiation has actually more properties, all the algebraic properties of `+`, × and `exp` should be considered to capture to whole power of an attacker.

Remark

This protocol is used to perform local transactions, hence an agent can not have several opened sessions at the same time.

7 Addition, multiplication and modular exponentiation

Secure Remote Password Protocol

Author(s): Thomas Wu 1998

Summary: The *Secure Remote Password* [Wu98] protocol is an authenticated key-exchange protocol designed to resist both passive and active network adversaries even when used with relatively short, human-memorizable password.

Protocol specification

```
C, Serv :     principal
s, a, b, u :  number
H :           number -> number
alias x = H(s,C,P)
alias v = exp(g,x)
alias B = v+exp(g,b)

   1.      C    ->   Serv   :    C
   2.    Serv   ->    C     :    s
  C computes x = H(s,C,P)
   3.    C   ->   Serv   :    A = exp(g,a)
  Serv computes B = v + exp(g,b)
```

```
 4.    Serv  ->  C   :    B,u
C computes S = exp((B - exp(g,x)),a + u × x)
 5.    C  ->  Serv  :    M1 = H(A,B,S)
Serv computes S = exp(A × exp(v,u),b) and verifies M1 = H(A,B,S)
 6.    Serv  ->  C   :    H(A,M1,S)
C verifies H(A,M1,S) and both Serv and C compute K = H(S)
```

Description of the protocol rules

All values are computed modulo a large, safe integer N and `g` is a primitive root. The client `C` starts the protocol by sending his name to the server. The server generates a fresh number `s`. Then the client and the server exchanges the value `A` and `B`. They are now able to both compute `S = exp(B - exp(g,x),a + u × x) = exp(A × exp(v,u),b)`. They verify that they share the same number by sending two hashes: `H(A,B,S)` and `H(A,M1,S)`. At the end of a successful run, both sides will share a secret session key `K`.

Requirements

The $+$ and $\times$ symbols denote respectively the addition and multiplication, and exp the modular exponentiation. The axioms listed below are the minimal ones required to perform the verification test at the last step of the protocol.

- x + y = y + x (E1)
- x + (y + z) = (x + y) + z (E2)
- (-x) + x = 0 (E3)
- x + 0 = x (E4)
- exp(exp(x, y), z) = exp (x, y × z) (E5)
- exp(x, y) × exp(x, z) = exp(x,y+z) (E6)
- x × y = y × x (E7)

Details of the of `S` performed by `C`:

```
    exp((B - exp(g,x)),a + u × x)
 =  exp((exp(g,x) + exp(g,b) - exp(g,x)),a + u × x)
 =  exp((exp(g,b),a + u × x))                    (E1,E2,E3)
 =  exp(g, b × (a + u × x))                      (E5)
 =  exp(g,(a + u × x) × b)                       (E7)
 =  exp(exp(g, a + u × x), b)                    (E5)
 =  exp(exp(g, a) × exp(g, u × x))               (E6)
 =  exp(exp(g, a) × exp(g, x × u))               (E7)
 =  exp(exp(g, a) × exp(exp(g, x), u))           (E5)
 =  exp(A × exp(v, u))
```

Since the modular exponentiation has actually more properties, all the algebraic properties of `+`, $\times$ and `exp` should be considered to capture to whole power of an attacker.

References

[BGW01] Nikita Borisov, Ian Goldberg, and David Wagner. Intercepting mobile communications: The insecurity of 802.11. In *Proceedings of the Seventh Annual International Conference on Mobile Computing and Networking (MOBICOM-01)*, pages 180–188, New York, 2001. ACM Press.

[Bla01] Bruno Blanchet. An efficient cryptographic protocol verifier based on prolog rules. In IEEE, editor, *14th IEEE Computer Security Foundations Workshop (CSFW-14)*, june 2001.

[Bul97] J Bull. The authentication protocol. *APM Report*, 1997.

[CGS97] Ronald Cramer, Rosario Gennaro, and Berry Schoenmakers. A secure and optimally efficient multi-authority election scheme. *Lecture Notes in Computer Science*, 1233:103+, 1997.

[CJ97] John Clark and Jeremy Jacob. A survey of authentication protocol literature : Version 1.0., November 1997.

[CKRT03] Yannick Chevalier, Ralf Küsters, Michael Rusinowitch, and Mathieu Turuani. An np decision procedure for protocol insecurity with xor. In *Proc. of 18th Annual IEEE Symposium on Logic in Computer Science (LICS '03)*, 2003.

[CLC03] H. Comon-Lundh and V. Cortier. New decidability results for fragments of first-order logic and application to cryptographic protocols. In *Proc. of the 14th Int. Conf. on Rewriting Techniques and Applications (RTA'2003)*, volume 2706 of *LNCS*, pages 148–164. Springer-Verlag, June 2003.

[CV02] Y. Chevalier and L. Vigneron. Automated Unbounded Verification of Security Protocols. In Ed Brinksma and Kim Guldstrand Larsen, editors, *14th International Conference on Computer Aided Verification, CAV'2002*, volume 2404 of *Lecture Notes in Computer Science*, pages 324–337, Copenhagen (Denmark), July 2002. Springer.

[DLMS99] N. Durgin, P. Lincoln, J. Mitchell, and A. Scedrov. Undecidability of bounded security protocols. In *Proc. of the Workshop on Formal Methods and Security Protocols*, 1999.

[DS81] D. Denning and G. Sacco. Timestamps in key distributed protocols. *Communication of the ACM*, 24(8):533–535, 1981.

[ElG85] T. ElGamal. A public-key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, IT-31:469–472, 1985.

[FPS00] Pierre-Alain Fouque, Guillaume Poupard, and Jacques Stern. Sharing decryption in the context of voting or lotteries. *Lecture Notes in Computer Science*, 2000.

[GP01] M. Girault and J.C. Paillès. Contactless EP: a public key solution with good performances. Unpublished, 2001.

[JRV00] F. Jacquemard, M. Rusinowitch, and L. Vigneron. Compiling and verifying security protocols. In *Logic for Programming and Automated Reasoning (LPAR'00)*, volume 1955 of *LNCS*, November 2000.

[L.M99] L.M.S.C of the IEEE Computer Science Society. *Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications*, 1999.

[Low95] Gavin Lowe. An attack on the needham-schroeder public key authentication protocol. *Information Processing Letters*, 56(3):131–136, november 1995.

[Low97] G. Lowe. Casper: A compiler for the analysis of security protocols. In *Proc. of 10th Computer Security Foundations Workshop (CSFW'97)*. IEEE Computer Society Press, 1997. Also in Journal of Computer Security, Volume 6, pages 53-84, 1998.

[Low98] G. Lowe. Towards a completeness result for model checking of security protocols. In *Proc. of the 11th Computer Security Foundations Workshop (CSFW'98)*. IEEE Computer Society Press, 1998.

[LR97] G. Lowe and A. W. Roscoe. Using CSP to detect errors in the TMN protocol. *Software Engineering*, 23(10):659–669, 1997.

[MGA03] Jacques Traoré Marc Girault and David Arditti. Les avancées de la cryptologie. In *20e Conférence France Télécom Recherche*, Paris, 2003. To appear.

[NS78] R. Needham and M. Schroeder. Using encryption for authentication in large networks of computers. *Communications of the ACM*, 21(12), December 1978.

[Pai99] P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. *jounral of Lecture Notes in Computer Science*, 1592:223–??, 1999.

[PQ00] Olivier Pereira and Jean-Jacques Quisquater. On the perfect encryption assumption. In *Proceedings of the Workshop on Issues in the Theory of Security (WITS 2000)*, pages 42–45, Geneva, Switzerland, July 2000.

[RS98] P. Ryan and S. Schneider. An attack on a recursive authentication protocol: A cautionary tale. *Information Processing Letters*, 65(1):7–10, 1998.

[RT01] M. Rusinowitch and M. Turuani. Protocol insecurity with finite number of sessions is NP-complete. In *Proc. of the 14th Computer Security Foundations Workshop (CSFW'01)*, pages 174–190. IEEE Computer Society Press, 2001.

[Sch96] B. Schneier. *Applied Cryptography*. John Wiley & Sons, New York, 1996.

[Sim94] Gustavus J. Simmons. Cryptoanalysis and protocol failure. *Communications of the ACM*, 37(11):56–65, November 1994.

[spo] Security Protocols Open Repository. `http://www.lsv.ens-cachan.fr/spore/index.html`.

[TMN89] M. Tatebayashi, N. Matsuzaki, and D.B. Newman. Key distribution protocol for digital mobile communication systems. In *Advance in Cryptology — CRYPTO '89*, volume 435 of *LNCS*, pages 324–333. Springer-Verlag, 1989.

[Wu98] Thomas Wu. The secure remote password protocol. In *Proceedings of the 1998 Internet Society Network and Distributed System Security Symposium*, pages 97–111, San Diego, CA, march 1998.