UNIVERSITA' DEGLI STUDI DI PARMA
Dipartimento di Ingegneria dell'Informazione

Cryptography: Entity Authentication

Luca Veltri

(mail.to: luca.veltri@unipr.it)

Course of Network Security, Spring 2014

http://www.tlc.unipr.it/veltri

Entity Identification (Authentication)

- Techniques designed to allow one party (the verifier or authenticator) to gain assurances that the identity of another (the prover, claimant, or supplicant) is as declared, preventing impersonation
- The most common technique is by the verifier checking the correctness of a message (possibly in response to an earlier message) which demonstrates that the claimant is in possession of a secret associated with the genuine party
- A major difference between entity authentication and message authentication is that
  - **message authentication itself provides no timeliness guarantees with respect to when a message was created**
  - **whereas entity authentication involves corroboration of a claimant's identity through actual communications with an associated verifier during execution of the protocol itself**

Objectives

- Objectives of an identification protocol:
  - **In the case of honest parties A and B, A is able to successfully authenticate itself to B, i.e., B will complete the protocol having accepted A's identity**
  - **B cannot reuse an identification exchange with A so as to successfully impersonate A to a third party D (no transferability)**
  - **The probability is negligible that any party C distinct from A, carrying out the protocol and playing the role of A, can cause B to complete and accept A's identity**
  - **The previous points remain true even if:**
    - a large number of previous authentications between A and B have been observed
    - the adversary C has participated in previous protocol executions with either or both A and B
    - multiple instances of the protocol, possibly initiated by C, may be run simultaneously
- Zero-knowledge-based protocol:
  - **is that protocol executions do not even reveal any partial information which makes C's task any easier whatsoever**

Remarks

- Identification protocols provide assurances only at the particular instant in time of successful protocol completion
  - **If ongoing assurances are required, additional measures may be necessary**

Basis of identification

- Entity authentication techniques may be divided into three main categories, depending on which of the following the security is based:
  - **1. something known**
    - e.g. standard passwords (sometimes used to derive a symmetric key), Personal Identification Numbers (PINs), and the secret or private keys
  - **2. something possessed**
    - this is typically a physical accessory, as a "passport"
    - e.g. magnetic-striped cards, chip cards (also called smart cards or IC cards), and hand-held customized calculators (passwd generators) which provide time-variant passwords
  - **3. something inherent to a human individual**
    - this category includes methods which make use of human physical characteristics and involuntary actions (biometrics), such as handwritten signatures, fingerprints, voice, retinal patterns, hand geometries, and dynamic keyboarding characteristics
    - these techniques are typically non-cryptographic and are not discussed further

Properties of identification protocols

- Identification protocols may the following properties:
  - **reciprocy of identification**
    - either one or both parties may corroborate their identities to the other, providing, respectively, unilateral or mutual identification
  - **computational efficiency**
    - the number of operations required to execute a protocol
  - **communication efficiency**
    - this includes the number of passes (message exchanges) and the bandwidth required (total number of bits transmitted)

Properties of identification protocols (cont.)

- More subtle properties include:
  - **real-time involvement of a third party (if any)**
    - e.g.: an on-line trusted third party to distribute symmetric keys to communicating entities for authentication purposes; or an on-line (untrusted) directory service for distributing public-key certificates
  - **storage of secrets**
    - e.g.: the location and method used to store critical keying material (e.g., software only, local disks, hardware tokens, etc.)

Authentication attacks

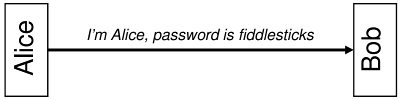
- Possible authentication attacks are:
  - **Impersonation attacks (pretend to be client or server)**
  - **Replay attacks (a valid message is copied and later resent)**
  - **Reflection attacks (re-send the authentication messages elsewhere)**
  - **Modify messages between client and server**
  - **Steal client/server authentication database**

Replay and reflection

- Countermeasures against replay and reflection attacks include
  - **one-way methods using of sequence numbers**
    - generally impractical
  - **one-way methods using timestamps**
    - needs synchronized clocks
  - **two-way methods using challenge/response**
    - using unique nonce, salt, realm values

Passwords

- Basic system uses passwords
- Conventional password schemes involve time-invariant passwords
  - **which provide so-called weak authentication**
  - **they serve as a shared secret between the user and system**
    - thus fall under the category of symmetric-key techniques providing unilateral authentication
- To gain access to a system resource the user enters a (userid, password) pair



- Password schemes are distinguished by the means by which information allowing password verification is stored within the system, and the method of verification

Password storing and verification

- Stored (cleartext) password files
  - **the most obvious approach is for the system to store user passwords cleartext in a system password file, which is both read- and write-protected**
  - **the system compares the entered password to the password file entry for the corresponding userid**
  - **this is classified as a non-cryptographic technique**
  - **A drawback of this method is that it provides no protection against privileged insiders**
  - **storage of the password file on backup media is also a security concern, since the file contains cleartext passwords**

Password storing and verification (cont.)

- "Encrypted" (or hashed) password files
  - **a one-way function of each user password is stored in place of the password itself**
  - **to verify a user-entered password, the system computes the one-way function of the entered password, and compares this to the stored entry for the stated userid**
  - **the password file need now only be write-protected**
  - **the use of a one-way function is generally preferable to reversible encryption**
    - in both cases, for historical reasons, the resulting values are typically referred to as "encrypted" passwords
  - **protecting passwords by either method before transmission over public communications lines addresses the threat of compromise of the password itself, but alone does not preclude disclosure or replay of the transmission**

Password storing and verification (cont.)

- Several possibilities for the place where password are maintained:
  - **user passwd individually stored into each host**
  - **host retrieve the passwd from one location (authentication storage node)**
  - **host send user's information to a authentication facilitator node (Authentication Server) that performs authentication and tells the response (e.g. yes/no)**
- Last two cases require a security association between the host and the authentication node

Password attacks

- Replay of fixed passwords
  - **a weakness of schemes sending fixed, reusable passwords is the possibility that an adversary learns a user's password by observing it as it is typed in**
  - **a second security concern is that user-entered passwords (or one-way hashes thereof) are transmitted in cleartext over the communications line**
    - an eavesdropping adversary may record this data, allowing subsequent impersonation
  - **fixed password schemes are thus of use only when the password is transmitted over trusted communications lines safe from monitoring**

Password attacks (cont.)

- Exhaustive password search
  - **an adversary simply (randomly or systematically) tries passwords, one at a time**
  - **this may be countered by:**
    - ensuring passwords are chosen from a sufficiently large space
    - limiting the number of invalid (on-line) attempts allowed within fixed time periods, and
    - slowing down the password mapping or login-process itself
  - **the feasibility of the attack depends on the number of passwords that need be checked before a match is expected and the time required to test each**
- Password-guessing and dictionary attacks
  - **to improve upon the expected probability of success of an exhaustive search, an adversary may search the space in order of decreasing (expected) probability**

Password attacks (cont.)

- On-line password guessing
  - **direct password search**
  - **defense/trick:**
    - maximum number of attempts
    - slow down
- Off-line password guessing
  - **the intruder captures a quantity derived by a passwd**
    - e.g. a challenge response, or a hash within a database
  - **does not require interacting with the actual verifier until final stage**
  - **arbitrary amount of power**

Countermeasures to password attacks

- Password rules
  - **Since dictionary attacks are successful against predictable passwords, some systems impose "password rules" to discourage or prevent users from using "weak" passwords**
  - **typical password rules include a lower bound on the password length (e.g., 8 or 12 characters)**
  - **a requirement for each password to contain at least one character from each of a set of categories (e.g., uppercase, numeric, non-alphanumeric)**
  - **or checks that candidate passwords are not found in on-line or available dictionaries, and are not composed of account-related information such as userids or substrings thereof**
  - **the objective of password rules is to increase the entropy (rather than just the length) of user passwords beyond the reach of dictionary and exhaustive search attacks**

Countermeasures to password attacks (cont.)

- Slowing down the password mapping
  - **To slow down attacks which involve testing a large number of trial passwords, the password verification function (e.g., one-way function) may be made more computationally intensive**
    - for example, by iterating a simpler function t>1 times
    - the total number of iterations must be restricted so as not to impose a noticeable or unreasonable delay for legitimate users

Countermeasures to password attacks (cont.)

- Salting passwords
  - **to make dictionary attacks less effective, each password, upon initial entry, may be augmented with a t-bit random string called a salt**
  - **both the hashed password and the salt are recorded in the password file**
  - **the difficulty of exhaustive search on any particular user's password is unchanged by salting**
    - since the salt is given in cleartext
  - **however, salting increases the complexity of a dictionary attack against a large set of passwords simultaneously**
    - requiring the dictionary to contain 2^t variations of each trial password
  - **note that with salting, two users who choose the same password have different entries in the system password file**

Countermeasures to password attacks (cont.)

- Passphrases
  - **to allow greater entropy without stepping beyond the memory capacity of human users, passwords may be extended to passphrase**
    - the idea is that users can remember phrases easier than random character sequences

PINs and passkeys

- Personal identification numbers (PINs) fall under the category of fixed (time-invariant) passwords
  - **they are most often used in conjunction with “something possessed” (a token), typically a physical token such as a plastic banking card with a magnetic stripe, or a chipcard**
  - **to prove one’s identity as the authorized user of the token**
  - **for user convenience PINs are typically short and numeric, e.g., 4 to 8 digits**
- To prevent exhaustive search through such a small key space additional procedural constraints are necessary
  - **e.g., ATMs confiscate a card if three incorrect PINs are entered successively, or cause the card to be “locked” or deactivated**
- In an off-line system without access to a central database, information facilitating PIN verification must be stored on the token itself

Password-derived keys

- Sometimes is useful to map a user password into a cryptographic key
  - **e.g. by using a one-way hash function**
  - **such password-derived keys are called passkeys**
- The passkey is then used to secure a communications link between the user and a system which also knows the user password
- Sometimes, conversion can be more tricky (and computationally expensive)
  - **due to key properties**
- It should be ensured that the entropy of the user’s password is sufficiently large that exhaustive search of the password space is not more efficient than exhaustive search of the passkey space

Example of Key derivation algorithm

- OpenSSL Key derivation algorithm
  - **The key and IV is derived by concatenating D_1, D_2, etc until enough data is available for the key and IV. D_i is defined as:**
  - **D_i = HASH^count(D_(i-1) || password || salt)**
  - **where || denotes concatentaion, D_0 is empty, HASH is the digest algorithm in use, HASH^1(data) is simply HASH(data), HASH^2(data) is HASH(HASH(data)) and so on**
  - **The initial bytes are used for the key and the subsequent bytes for the IV**
  - **Increasing the count parameter slows down the algorithm which makes it harder for an attacker to perform a brute force attack using a large number of candidate passwords.**

One-time passwords

- A major security concern of fixed password schemes is eavesdropping and subsequent replay of the password
- A partial solution is one-time passwords: each password is used only once
  - **such schemes are safe from passive adversaries who eavesdrop and later attempt impersonation**
- Can be implemented in Smart/token Cards

One-time passwords (cont.)

- Some one-time passwords variations:
  - **shared lists of one-time passwords**
    - use a sequence or set of secret passwords, (each valid for a single authentication), distributed as a pre-shared list
    - a drawback is maintenance of the shared list
    - if the list is not used sequentially, the system may check the entered password against all remaining unused passwords
    - a variation involves use of a challenge-response table
  - **sequentially updated one-time passwords**
    - during authentication using password *i*, the user creates and transmits to the system a new password (password *i+1*) encrypted under a key derived from password *i*
    - this method becomes difficult if communication failures occur
  - **one-time password sequences based on a one-way function**
    - more efficient than sequentially updated one-time passwords
    - may be viewed as a challenge-response protocol where challenge is implicitly defined by the current position within the pwd sequence

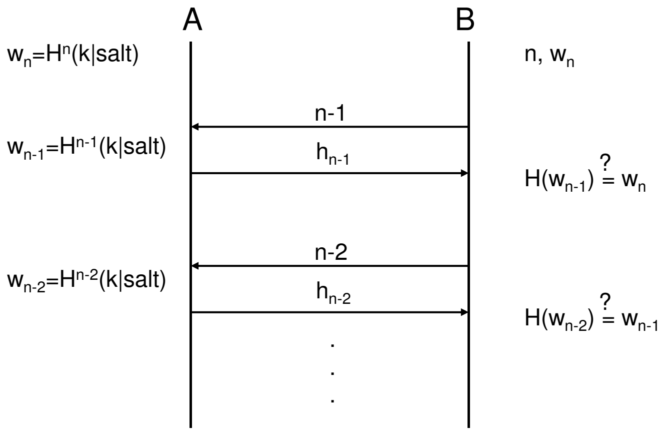
OTPs based on one-way functions (Lamport's scheme)

- The user begins with a secret w; a one-way function H is used to define the password sequence (used in reverse order):
  - **w, H(w), H(H(w)), ... , H^t(w)**
- The password for the i^{th} identification session, $1 \le i \le t$, is:
  - **$w_i = H^{t-i}(w)$**
- Lamport's one-way-function(OWF)-based one-time passwords:
  - **user A begins with a secret w**
  - **a constant t is fixed (e.g., t = 100 or 1000), defining the number of identifications to be allowed**
  - **A transfers (the initial shared secret) w0 = H^t(w), in a manner guaranteeing its authenticity, to the system B**
  - **B initializes its counter i_A for A to 1 ($i_A = 1$)**
  - **A → B : A, i, $w_i = H^{t-i}(w)$**
    - A's equipment computes $w_i = H^{t-i}(w)$ (easily done either from w itself, or from an appropriate intermediate value saved during the computation of H^t(w) initially), and transmits it to B
  - **B checks that $i = i_A$, and that the received password w_i satisfies: $H(w_i) = w_{i-1}$**

SKey

- **Sistema per la generazione di password dinamiche**
- **Al login, all'utente viene inviato un seme per la generazione della password**
- **L'utente esegue localmente (es. sul suo host) la generazione della password (in funzione del seme inviato) e la comunica al server**
- **Il server confronta quanto ricevuto con la propria password e, se vi è coincidenza, autentifica l'utente**

SKey



A

B

$w_n=H^n(k|salt)$

n, w_n

n-1

$w_{n-1}=H^{n-1}(k|salt)$

h_{n-1}

$H(w_{n-1}) \overset{?}{=} w_n$

n-2

$w_{n-2}=H^{n-2}(k|salt)$

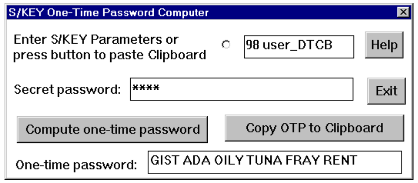
h_{n-2}

$H(w_{n-2}) \overset{?}{=} w_{n-1}$

.
.
.

Esempio SKey

```
>telnet 193.205.102.131
Trying 193.205.102.131 ...
Connected to 193.205.102.131.
Escape character is '^]'.
Servizio TELNET - Firewall
.........................
Inizio sessione:
CheckPoint FireWall-1 authenticated Telnet server
Login: user_DTCB
SKEY CHALLENGE: 98 user_DTCB
Enter SKEY string: GIST ADA OILY TUNA FRAY RENT
User user_DTCB authenticated by S/Key system.
```



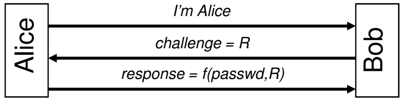
Enter S/KEY Parameters or press button to paste Clipboard: 98 user_DTCB

Secret password: ****

One-time password: GIST ADA OILY TUNA FRAY RENT

Challenge-response identification

- The idea is that one entity (the claimant) "proves" its identity to another entity (the verifier) by demonstrating knowledge of a secret, without revealing the secret itself to the verifier
  - **in some mechanisms, the secret is known to the verifier, and is used to verify the response**
  - **in others, the secret need not actually be known by the verifier**
- This is done by providing a response to a *time-variant* challenge, where the response depends on both the entity's secret and the challenge
  - **the challenge is typically a number (or a bit string) chosen by one entity (randomly and secretly) at the outset of the protocol**
- *Time-variant* parameters may be used to counteract replay and interleaving attacks



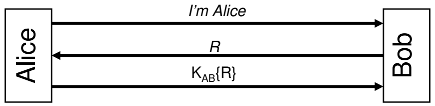
Challenge-response identification (cont.)

- Time-variant parameters:
  - **nonces (or random numbers)**
    - a value (a random number, a random string or random bit string) used no more than once for the same purpose
    - it typically serves to prevent (undetectable) replay
  - **sequence numbers**
    - a sequence number (serial number, or counter value) serves as a unique number identifying a message (or authentication instance)
    - typically used to detect message replay
    - a message is accepted only if the sequence number therein has not been used previously
    - the simplest policy is that a sequence number is incremented sequentially
    - a less restrictive policy is that sequence numbers need (only) be monotonically increasing
      - this allows for lost messages due to non-malicious communications errors, but precludes detection of messages lost due to adversarial intervention

Challenge-response identification (cont.)

- Used values: (cont.)
  - **timestamps**
    - timestamps may be used to provide timeliness and uniqueness guarantees, to detect message replay
    - they may also be used to implement time-limited access privileges, and to detect forced delays
    - a message is valid if:
      - the timestamp difference is within an acceptance window (a fixed-size time interval, e.g., 10 milliseconds or 20 seconds)
      - (optionally) no message with an identical timestamp has been previously received from the same originator
    - timestamps in protocols offer the advantage of fewer messages (typically by one), and no requirement to maintain pairwise long-term state information (cf. sequence numbers)
    - the main drawback of timestamps is the requirement of maintaining secure, synchronized distributed timeclocks

Authentication with symmetric key



Alice

Bob

I'm Alice

knows alice's secret

R

Picks random R

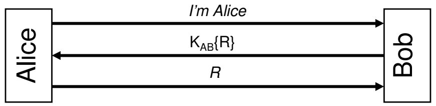
computes $K_{AB}\{R\}$

$K_{AB}\{R\}$

computes and compare the $K_{AB}\{R\}$

- Note:
  - **does not require reversible cryptography**
    - examples: $K_{AB}\{R\}=E(K_{AB},R)$, $K_{AB}\{R\}=MAC(K_{AB},R)$
- drawbacks:
  - **authentication is not mutual**
  - **an eavesdropper could mount an off-line password guessing attack**
  - **some who read the Bob's passwd-database can later impersonate Alice**

Authentication with symmetric key (variant 1)



Alice

Bob

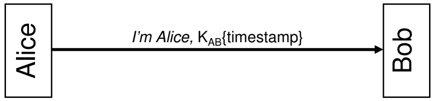
I'm Alice

$K_{AB}\{R\}$

R

- differences:
  - **requires reversible cryptography**
  - **if R is a recognizable quantity with limited lifetime (e.g. a random number concatenated with a timestamp), Alice can authenticate Bob**
  - **if R is a recognizable quantity, Carol can mount an offline passwd-guessing attack without eavesdropping**
    - Carol obtains $K_{AB}\{R\}$ (second message) by just sending the first message to B claiming to be A

Authentication with symmetric key (variant 2)

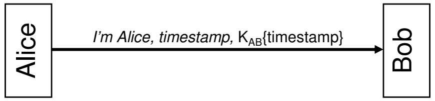


Alice

Bob

I'm Alice, $K_{AB}\{timestamp\}$

- differences:
  - **this mechanism can be added very easily to a protocol designed for cleartext passwd sending**
  - **more efficent**
  - **several pitfalls due to the time validity (time synchronization between Alice and Bob, authentication with multiple server with the same passwd, etc)**
- variant:



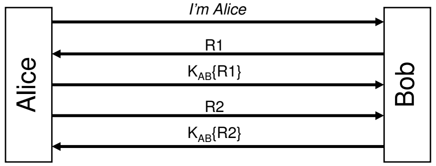
Alice

Bob

I'm Alice, timestamp, $K_{AB}\{timestamp\}$

  - **$K_{AB}\{.\}$ does not require to be reversible**

Mutual authentication with symmetric key



Alice

Bob

I'm Alice

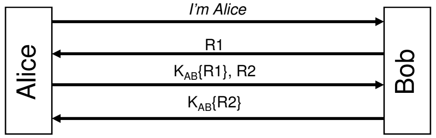
R1

$K_{AB}\{R1\}$

R2

$K_{AB}\{R2\}$

- or shorter..



Alice

Bob

I'm Alice

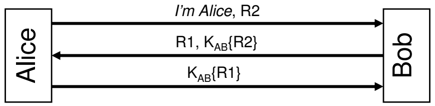
R1

$K_{AB}\{R1\}$, R2

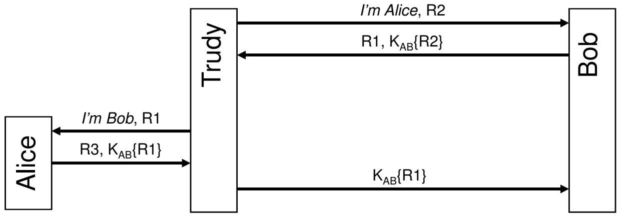
$K_{AB}\{R2\}$

Mutual authentication with symmetric key

or shorter..

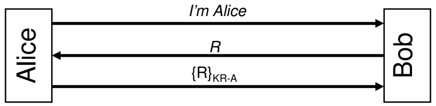


- but:
  - **Reflection attack**

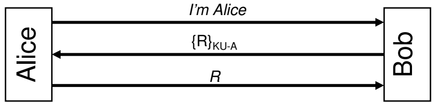


- Good general principle of security protocol:
  - **the initiator should be the first to prove its identity**

Authentication with private/public key

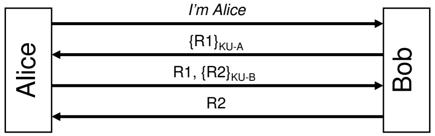


or

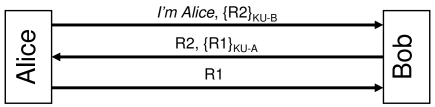


- property:
  - **the database at Bob is no-longer security-sensitive**
    - must be protected (only) for unauthorized modification, but not from reading
      - only authentication and integrity protection
- drawback:
  - **if you can trick Alice into signing something, you can impersonate Alice**
- countermeasures:
  - **not use the same key for two different purpose unless the design for all uses are coordinated (this is a general rule!), and/or**
  - **impose enough structure to be signed (nounce, realm, timestamp, etc.)**

Mutual authentication with private/public key



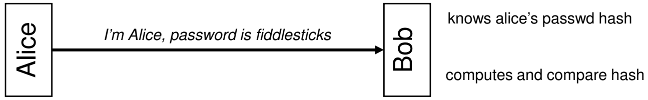
or shorter..


Alice
Bob
I'm Alice, $\{R2\}_{KU\text{-}B}$
R2, $\{R1\}_{KU\text{-}A}$
R1

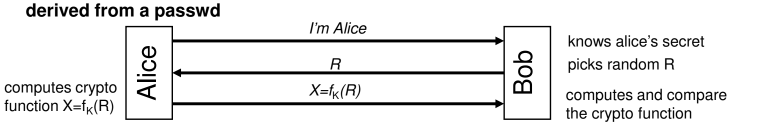
- Public-key issues:
  - **how obtaining public key of the peer-entity**
  - **how storing public key of the peer-entity**
  - **how storing own private key**

Eavesdropping and server database reading

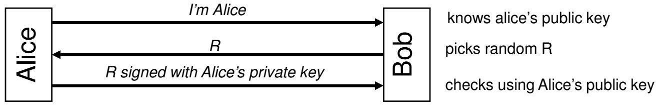
- Protection against server database reading:
  - **vulnerable to eavesdropping**


Alice
Bob
I'm Alice, password is fiddlesticks
knows alice's passwd hash
computes and compare hash

- Protection against eavesdropping:
  - **vulnerable to database reading, and to offline password guessing if the secret (key) is derived from a passwd**


Alice
Bob
I'm Alice
R
X=f$_K$(R)
computes crypto function X=f$_K$(R)
knows alice's secret
picks random R
computes and compare the crypto function

- Protection against both using asymmetric cryptography:


Alice
Bob
I'm Alice
R
R signed with Alice's private key
knows alice's public key
picks random R
checks using Alice's public key

Zero-knowledge identification protocols

- Zero-knowledge (ZK) protocols allow a prover to demonstrate knowledge of a secret while revealing no information whatsoever
  - **beyond what the verifier was able to deduce prior to the protocol run**
- ZK protocols are often instances of interactive proof system, wherein a prover and verifier exchange multiple messages (challenges and responses), typically dependent on random numbers (ideally: the outcomes of fair coin tosses) which they may keep secret
  - **the prover's objective is to convince (prove to) the verifier the truth of an assertion, e.g., claimed knowledge of a secret**
  - **the verifier either accepts or rejects the proof**

Example of zero-knowledge proof: Fiat-Shamir identification protocol

- Basic version of the Fiat-Shamir protocol
  - **the objective is for A to identify itself by proving knowledge of a secret *s* to any verifier B, without revealing any information about *s*, not known or computable by B prior to execution of the protocol**
  - **the security relies on the difficulty of extracting square roots modulo large composite integers *n* of unknown factorization, which is equivalent to that of factoring *n***
- Protocol
  - **a trusted center T selects and publishes an RSA-like modulus $n = pq$, keeping primes *p* and *q* secret**
  - **each claimant A selects a secret *s* coprime to *n*, $1 \leq s \leq n - 1$, computes $v = s^2$ mod *n*, and registers v with T as its public key**
  - **each of *t* rounds has three messages with form as follows**
    - A → B : $x = r^2 \bmod n$ (witness)
    - A ← B : $c \in \{0,1\}$ (challenge)
    - A → B : $y = r\ s^c \bmod n$ (response)

Example of zero-knowledge proof: Fiat-Shamir identification protocol (cont.)

- Verification:
  - **if c=0, B verifies that $y^2=x \bmod n$**
  - **if c=1, B verifies that $y^2=xv \bmod n$**
- Explanation:
  - **the challenge (or exam) *c* requires that A be capable of answering two questions, one of which demonstrates her knowledge of the secret *s*, and the other an easy question (for honest provers) to prevent cheating**
  - **an adversary impersonating A might try to cheat by selecting any *r* and setting $x = r^2/v$, then answering the challenge $c = 1$ with a "correct" answer $y = r$; but would be unable to answer the exam $c = 0$ which requires knowing a square root of *x mod n***
  - **a prover A knowing *s* can answer both questions, but otherwise can at best answer one of the two questions, and so has probability only 1/2 of escaping detection**
  - **to decrease the probability of cheating arbitrarily to an acceptably small value of $(1/2)^t$ (e.g., $t = 20$ or $t = 40$), the protocol is iterated *t* times, with B accepting A's identity only if all *t* questions (over *t* rounds) are successfully answered**