Password-based Authentication and Authenticated Key Exchange

Shouhuai Xu
Laboratory for Information Security Technology
www.list.gmu.edu/~shxu

Password-based A&AKE

- Background knowledge
- Previous solutions to authentication: The Trouble
- Being implemented proposals: Server has a cert.
- Weaker assumption proposals: Server has no cert.
- Password protocols and public key cryptography
- Future directions

Semantic Security of PKC

- Intuition: Whatever is computable about the plaintext of a given ciphertext is computable without the ciphertext. (Analogy to Shannon's perfect secrecy of the plaintext!)
- Definition. Adversary A receives a challenge ciphertext $c \in_R ENC(m)$ where $m \in_R M$ and produce x. Compare it with that A' does not receive ENC(m) and produces x'. We say that ENC is semantic security, if for all PPT (Probabilistic polynomial time) relations R(m, x) and all PPT A there exists a PPT A' such that

  $$|Pr[R(m, x)] - Pr[R(m, x')]| < 1/poly(k).$$

- The outputs of A and A' are indistinguishable!
- Alternative definition: Challenge the adversary with a ciphertext $c \in_R ENC(m)$ where $m \in_R ENC(m_0, m_1)$. Ask the adversary to guess which plaintext is used? The probability for the adversary to win is negligibly more than ½.

Non-malleable Security of PKC

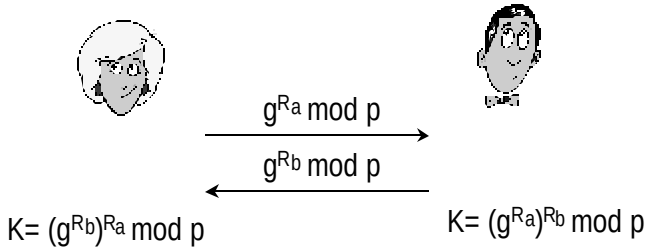
- Intuition: Whatever is computable in an encrypted form about the plaintext of a given ciphertext is computable without the ciphertext. (Still analogy to perfect secrecy, but not for plaintext directly!)
- Definition [DDN91]. Adversary A receives a challenge ciphertext $c \in_R ENC(m)$ where $m \in_R M$ and produce x. Compare it with that A' does not receive ENC(m) and produces x'. We say that ENC is non-malleable if for all PPT relations R(m, x) and all PPT A there exists a PPT A' such that

  $$|Pr[R(m, y)] - Pr[R(m, y')]| < 1/poly(k).$$

  Where y is the decryption of x and y' is the decryption of x'.
- The decryption of the outputs of A and A' are indistinguishable!

DH Key Exchange

- Basic Diffie-Hellman: unauthenticated key exchange
- p : large prime, g : generator of certain group



ROM

- A hash function f takes a variable length input and outputs a fixed length “fingerprint”
- A good hash function is usually *one-way* and *collision resistant*
- Sometimes we assume a hash function behaves like a random oracle – Random Oracle Model
- Assume hash function (like SHA used in a special way) behaves like a totally random function
- Used for proving security of many practical protocols, including OAEP (RSA PKCS #1 v.2.0)

ICM

Ideal Cipher Model:

Given $k \in \{0,1\}^*$, we set ENC_k to be a random one-to-one function, and we let ENC_k be defined by $DEC_k(y)$ is the value x such that $ENC_k(x)=y$, and BAD otherwise.

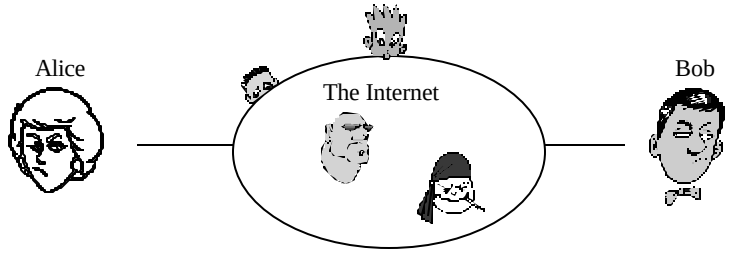
Note: The ideal cipher model is an even richer assumption than Random Oracle model. While it may be constructed from Random Oracle in an trivial way, it is still an open problem to formally support this.

PRF

- Pseudo-random function (PRF): A keyed function (or "function family") whose outputs look random if the key is unknown.
- Example
  - Let f be a PRF. Let k be a random key.
  - Let x be a public value. If k is unknown, $f_k(x)$ is indistinguishable from real randomness.

- Background knowledge
- Previous solutions to authentication: The Trouble
- Being implemented proposals: Server has a cert.
- Weaker assumption proposals: Server has no cert.
- Password protocols and public key cryptography
- Future directions

Authentication



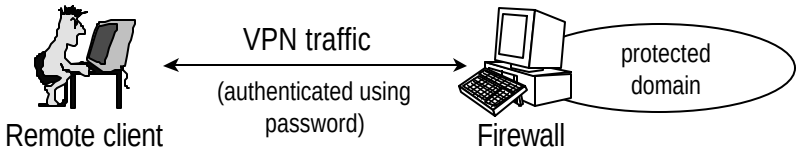
- In the real world we know "who's who" via various methods. (Perhaps the most effective way is that, Aha, we had ever met...)
- Well, how do we know "who's who" in the networked world?

Authentication

- "who's who in the networked world" is always established by one or combination of:
  - something a user has (smartcard/token)
  - something a user is (fingerprint/voice scan)
  - something a user knows (password/short secret)

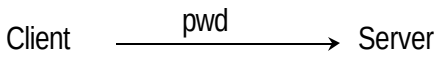
A Typical Scenario

- Remote user access
- Goal: The cost/overload on the user should be as little as possible

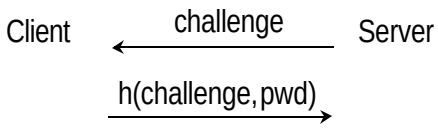


Previous Solutions

- Original Telnet - vulnerable to eavesdropping



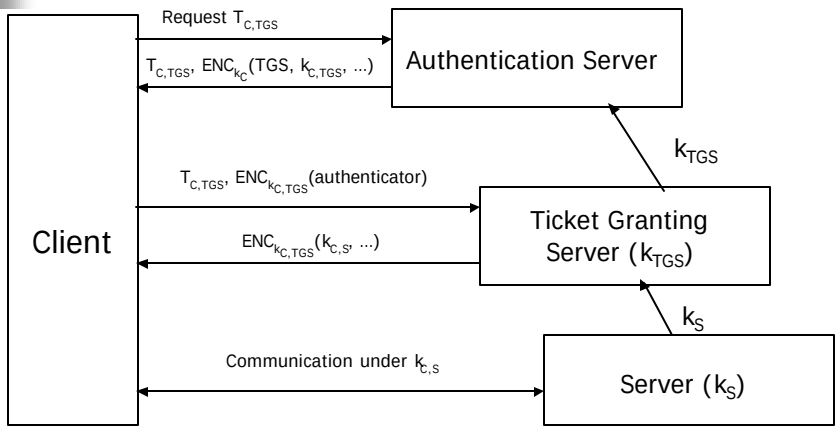
- Challenge-Response



The Trouble

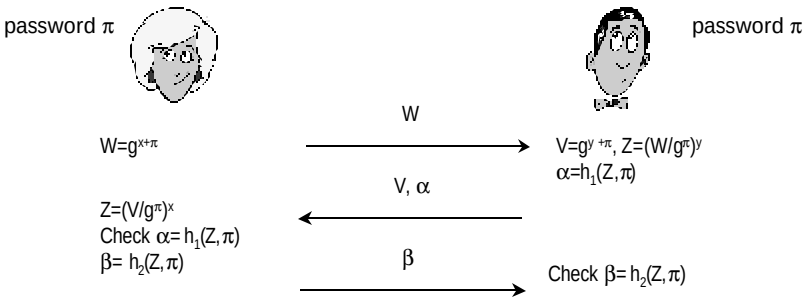
- Password is always weak (i.e., low entropy, easily guessed, drawn from a small dictionary)
- Dictionary attack
  - Given data generated using a password, can guess possible passwords from a dictionary and verify against the data d
  - Example: We know the challenge and the function h, we can test our guess of password∈ dictionary by checking that d=h(challenge, password)
  - It works no matter what function h is and, more importantly, it can be done offline!

The Trouble in Kerberos



The trouble: k_C is defined to be some one-way function of password!

A Subtle Example

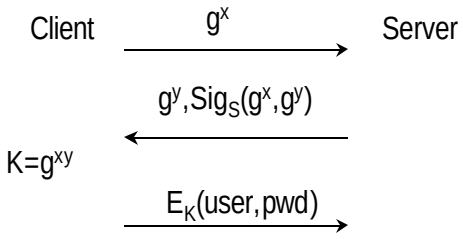


- Try to find the hole!

- Background knowledge
- Previous solutions to authentication: The Trouble
- Being implemented proposals: Server has a cert.
- Weaker assumption proposals: Server has no cert.
- Password protocols and public key cryptography
- Future directions

Secure SHell (SSH)

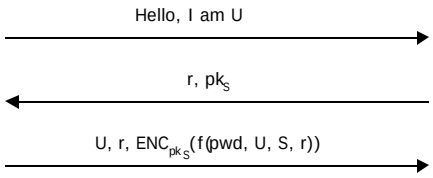
- SSH - Relies on public key



We don't focus on this protocol. Instead, we take a somewhat detailed look at [HK98, HK99, B99].

HK98

User (pwd) Server (pwd; pk_S)



- Is this protocol secure? (Assume that both f(pwd; ·) and f(·; x) are one-to-one. Actually, f(pwd; ·) can even be collision-resistant.)
- Intuitively, if we assume the user can verify the validity of the server's public key certificate, it is secure.
- However, such an intuition is not true even if the ENC is semantic secure!

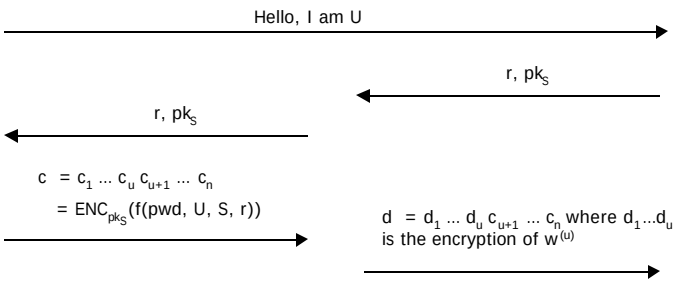
Semantic Security Is Not Enough

[HK98] gives an attack with $\log_{3/2}|\text{Dictionary}|$ attempts. The attacks goes as follows:

1. The sever sends random challenge r.
2. The attacker choose m passwords from the dictionary and for each password it computes f(pwd, U, S, r). Denote the resulting set by $W = \{w_1, ..., w_m\}$. Let $n = \max\{|w|: w \in W\}$.
3. Denote $w^{(i)}$ the i-bit string that appears most frequently as a prefix of the strings in W. Let p_i the probability of the strings in W having prefix $w^{(i)}$. For example, if w_1=01001, w_2=10111, w_3=1010, then n=5, $w^{(1)}$='1', $w^{(2)}$='10', etc. and p_1=2/3, p_i=2/3, etc. Clearly, $p_1 \geq 1/2$, $p_i \geq p_{i+1} \geq p_i/2$. Moreover, $p_n < 1/2$ (since f(·; x) is one-to-one). Therefore, there must exist some u such that $1/3 < p_u < 2/3$.
4. The attacker intercepts the user's response $c = ENC_{pk_S}(f(pwd, U, S, r))$ and substitutes it to get d such that
   $c = c_1 \ldots c_u c_{u+1} \ldots c_n$
   $d = d_1 \ldots d_u c_{u+1} \ldots c_n$
   where $d_1 \ldots d_u$ is the encryption of $w^{(u)}$.
5. If server accepts, the attacker knows that the first u-bit prefix is $w^{(u)}$, otherwise, it is not. Anyway, the size of the dictionary is shrunk. In the extreme case that m is the size of the dictionary, then at least 1/3 passwords are eliminated. So, at the worst $\log_{3/2}$m attempts is enough. (We can also use small m and then apply some standard probabilistic evaluation like Chernoff bounds to find the high successful probability for the attacker.)

Semantic Security Is Not Enough

User (pwd) Attacker Server (pwd; pk_S)



An even simplified scenarios is to substitute the encryption of c_1 with the encryption of 0.

How HK98 is flawed?

- One-ciphertext verification attack [HK98]:
  - The key generation algorithm GEN(k) outputs a pair of (pk, sk)
  - Given pk, the adversary generates a plaintext s
  - Let $r \in_R \{0,1\}^{|s|}$, $z \in_R \{s, r\}$, and the adversary is given $c=ENC_{pk}(z)$.
  - The adversary is allowed to generate a query (x', c'), where $c' \neq c$, whether or not $x'=DEC_{sk}(c')$
  - The adversary guesses whether $s=DEC_{sk}(c)$.
- Definition [HK98]. An encryption scheme (GEN, ENC, DEC) is said to resist one-ciphertext verification attacks if for any PPT adversary A:

$$|Pr[A \text{ guesses "encryption of s"} \mid s= DEC_{sk}(c)]$$
$$-Pr[A \text{ guesses "encryption of s"} \mid r= DEC_{sk}(c)]| \leq \varepsilon(k)$$

where $\varepsilon(k)$ is negligible.

How HK98 is flawed?

Theorem [B99]. The following scheme is resistant to one-ciphertext verification attack.

GEN. On input k and n, it outputs (e_1^0, d_1^0), (e_1^1, d_1^1), ..., (e_n^0, d_n^0), (e_n^1, d_n^1). Generates $h \in_R H$, a family of universal hash functions. (Such a family has the mathematical property that, for any x and a randomly chosen $h \in_R H$, to find $y \neq x$ such that $h(x)=h(y)$ is intractable.) The public encryption key is $(h, e_1^0, e_1^1, ..., e_n^0, e_n^1)$, and the corresponding private key is x $(d_1^0, d_1^1, ..., d_n^0, d_n^1)$.

ENC. To encrypt a k-bit message $m=b^1 ... b^k$.

- Select random bits b_j^i for $1 \le i \le k$, $1 \le j \le n$ such that $\oplus^n_{j=1} b_j^i$ for all $1 \le i \le k$ (i.e., each bit of the plaintext is stretched to n-bit).
- Generate a one-time signature key pair (with security parameter n). We denote F the verification key and P the signing key.
- Compute h(F). Denote the outputs $v=v_1...v_n$, where $v_i \in \{0,1\}$.
- For each $1 \le i \le k$ and $1 \le j \le n$, generate the ciphertext c_j^i of plaintext bit b_j^i using public key $e_j^{v_j}$.
- Generate a one-time signature s for the ciphertext c_j^i for $1 \le i \le k$ and $1 \le j \le n$.

How HK98 is flawed?

DEC. To decrypt $(F, s, c_1^1, ..., c_n^1, ..., c_1^k, ..., c_n^k)$.

- Using the one-time signature verification key F check the signature on the ciphertext.
- Compute h(F). Denote the outputs $v=v_1...v_1$, where $v_i \in \{0,1\}$.
- For each $1 \le i \le k$ and $1 \le j \le n$, decrypt the ciphertext c_j^i using private key $d_j^{v_j}$ to obtain plaintext bit b_j^i.

Attack the password system based on the one-verification ciphertext secure encryption. Assume that the adversary has corrupted a user A. Now we show how it can obtain the password of another uncorrupted user B. Denote the ciphertext $(F, s, c_1^1, ..., c_n^1, ..., c_1^k, ..., c_n^k)$ of $c = ENC_{pk_S}(pwd, B, S, r)$ (i.e., we assume that f is the concatenation function).

- The adversary authenticates himself as A. He generates a one-time signature key F' such that the jth bit of h(F') equals to the jth bit of h(F). (This is true for half of those F'.)
- Substitute the d_j^i (in A's reply) with c_j^i.
- If the server accepts, then the adversary knows b_j^i (of honest B's). After kn attempts, the adversary knows B's password. (The intrusion detection parameters is reset to zero!)

Remark

- The revision [HK99] of [HK98] is still criticized by [B99] though the encryption scheme of "one-ciphertext verification security" is changed to the security of "chosen ciphertext verification security".
- The potential attack: It is still possible to construct secure protocol under [HK99]'s definition that leaks the information that the passwords of two different un-corrupted passwords are equal. Therefore, the on-line attempts of the password guessing can be amortized to, possibly, bypass the intrusion detection system. Though this attack may not be so easy to impose in the real world, we should stress what we are concerned is whether the formal model is precise enough!
- According to [Hugo 2000], if the ENC is replaced with OAEP/CS98 scheme, then it is provably secure (authentication). Right now, I know it is easy to prove in the Definitional approach! (How about in the simulate model?)

More Robust Definition [B99]

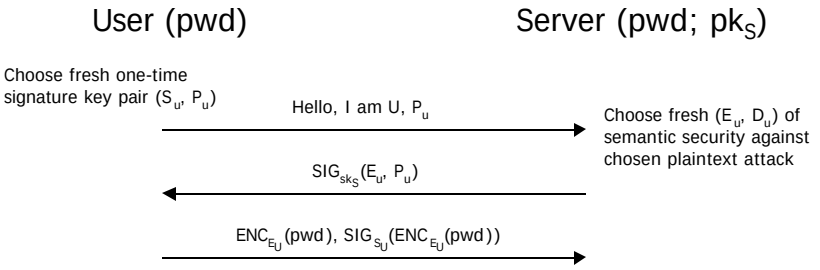
- The users in the system choose their password according to some joint distribution

$$D = (D_1, D_2, ..., D_{|U|})$$

where U is the universe of users. User u gets his password from distribution D_u.

- Definition [B99]. A protocol is secure against a given adversary class, if for all joint distribution D and all adversaries A in the given class there is a probabilistic polynomial time transcript simulator S that interacts with A, a password verification oracle PV, and a tape-writer T, such that the annotated transcripts of real long-lived runs with adversary A are probabilistic polynomial time indistinguishable from those produced by S.
- There are no two worlds!!!!!!

More Robust Construction [B99]



- Since each session uses different encryption key, semantic security is enough.

The Behavior of the Adversary

The adversary:

- The adversary may listen to any number of successful login attempts (challenges and encrypted responses) by other users
- The adversary can initiate any number of authentications, posing either as non-compromised users or compromised users
- The adversary can intercept and alter messages and generate spurious messages, in any direction between the server and any user, and it can learn if attempted authentications were successful
- The adversary has complete control over timing of events in the system.

How to model

- Ideally, we would like to say that the system is secure if the adversary cannot make the server accept a session not initiated by said user, just like the matching conversation. Unfortunately, such a definition is reasonable only if the password is cryptographic strong.
- Intuitively, we can say that the system is secure if the transcripts of the login session can be simulated without access to the password file. In this case, we have to use a password verification oracle to incorporate the on-line guessing of the adversary. Unfortunately, the simulator can use the password verification oracle to brute-force the password. Consequently, the simulator can simulate the success of the adversary's successful attack.
- Therefore, we have to define a transcript to include certain immutable records of the execution history; these will not be under the control of the simulator. In particular, queries to the password verification oracle, and attempts by non-compromised users to log in, will be recorded even during the simulation, and simulator is not allowed to alter these records. Intuitively, **this forces the simulator not to access the password oracle more frequently than the adversary tries to impersonate u.**

How to Model

- Each non-compromised user u is equipped with a special tape. Whenever the adversary causes u to initiate a login attempt, this fact is recorded on the tape. (successful: ε)
- The server is equipped with a special tape. Every time a (possibly impersonated) user attempts to log in, a record of the user being accessed, together with a bit saying whether or not the attempt was successful. (successful Σ, **if $S > \varepsilon$, then attacker succeeds**.)
- The annotated transcript of an execution includes all the messages exchanged between servers and (compromised or non-compromised) users, together with the contents of the special tapes. The intuition is that if a break occurs it will be reflected in the special tapes: the server's special tape will show more successful accesses to a user's account than attempts recorded on the user's special tape.
- In the simulation, the non-compromised users are our (i.e., simulator's) oracle in answering the first $real_m$ challenges. Of course, the simulator has access to the passwords of compromised users, since these are known to the adversary.

The Simulator

- The simulator will choose a secret/public pair of keys for the server. The simulator will choose a fixed random string σ. Whenever the adversary schedules a non-compromised user to respond to a challenge, the simulator will send a random encryption of σ.
- The simulator handles ciphertexts generated by the adversary different from those by non-compromised users.
  - The response ciphertext is generated by the attacker. We decrypt it and check the syntactical validity of the plaintext (i.e., pwd,u,S). If not, record an unsuccessful tuple in the special tape. If okay, ask the password verification oracle if the pwd is correct. The result is recorded on the special tape of the server.
  - The response ciphertext is generated by a non-compromised user. There are subcases:
    - ✓It is a genuine login by u (though activated by the attacker). The simulator requests the tape writer to record a successful login session.
    - ✓The attacker makes a compromised u' to make use of u to impersonate u. The simulator ask the tape writer to record a successful tuple iff there is server partner instance for u. Otherwise it is a failure record since $u' \neq u$.

The Reduction

- Assume there exists a distinguisher D. Let Q(k) be the running time of D on security parameter k. So, Q(k) is also the upper bound for the transcripts (thereof the running time of the simulator). Consequently, we have

$$|\Pr[D(\text{simulated view}) = 1] - \Pr[D(\text{real view})] = 1| \geq 1/\text{poly}(k)$$

- Given such an adversary, we can break the semantic security of the encryption scheme.
- $\text{real}_{Q(k)}$. This is the real transcripts.
- real_m for $m \in \{1,\ldots,Q(k)\}$. The first m challenges to non-compromised users are answered as exactly in the real world (i.e., via oracle query). After that, every reply by a non-compromised user is the encryption of a random string. (The adversary cannot distinguish them.)
- real_0. No challenges to non-compromised users are answered with proper encryption. Instead, all are encryption of a random string. It is identical in the real and in the simulated transcripts.
- Therefore, for some $m \in \{1,\ldots,Q(k)\}$ we have

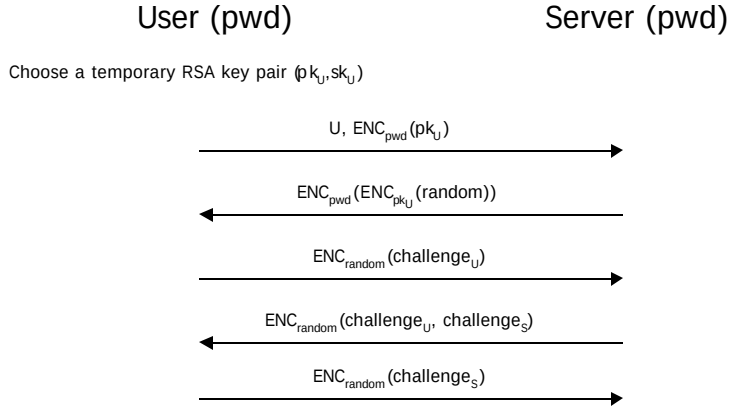
$$|\Pr[D(\text{real}_{m-1}) = 1] - \Pr[D(\text{real}_m)] = 1| \geq 1/[\text{poly}(k) \cdot Q(k)]$$

The Reduction

- We guess this m.
- At the m-th challenge, we replace the encryption of (pwd; U, S) with an encryption of a random element of {(pwd; U, S), σ} (i.e., we embedding the trap).
- If the encryption is for (pwd; u, S), we obtain a transcript of $real_{m-1}$, otherwise $real_m$.
- We present the transcript to D.
- We output D's answer.

- ☞Background knowledge
- ☞Previous solutions to authentication: The Trouble
- ☞Being implemented proposals: Server has a cert.
- ☞Weaker assumption proposals: Server has no cert.
- ☞Password protocols and public key cryptography
- ☞Future directions

EKE: RSA version [BM92]

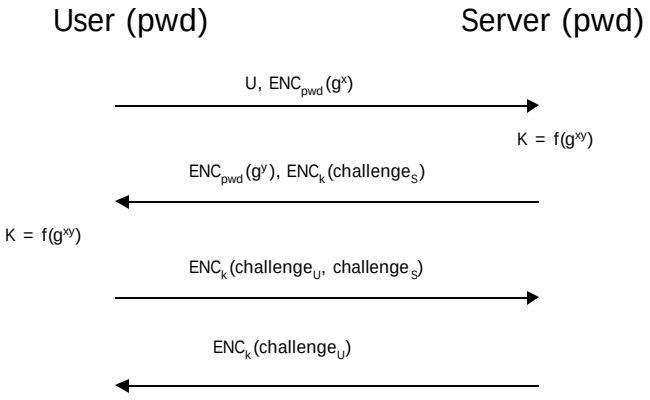


EKE: RSA version [BM92]

Potential problems:

- It is not clear how to encode efficiently a pair <e,n> so that it is indistinguishable from a random string. For example, n should not have small factors. Therefore, we may have to encode e only.
- Moreover, since e is always odd, we can overhear more valid sessions and the possible passwords can be narrowed down to one at a logarithm rate (This is the very difference from the Archot idea). If we don't encrypt the public key, the story is very different.
- The solution in [BM92] is to randomly add one to e before encrypting it with pwd. The user at the other end can remove the one if it was added because the users know e should always be odd.
- Information leakage can also result from fitting numbers of maximum size n into a block of size 2^m because a trial pwd generating a decryption greater than n is rejected.
- [BM92] said that an attacker even after substituting his own n for RSA modulus will not be able to mount a dictionary attack. [Patel, S&P97] presented such an attack.
- All simple modifications will eventually fail [Patel, S&P97]!

EKE: DH version [BM92]

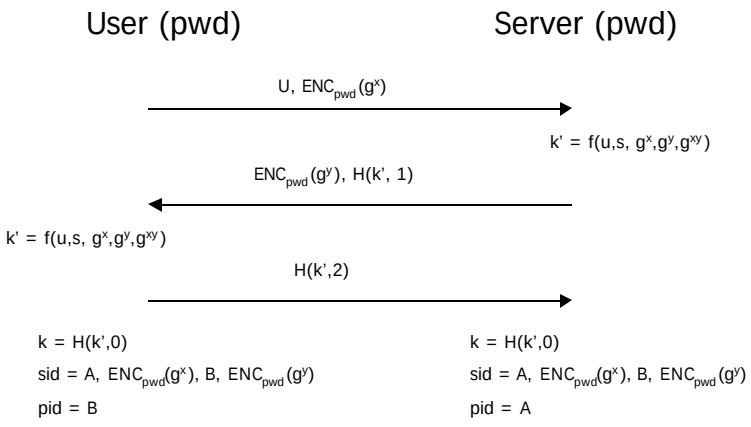


EKE: DH version [BM92]

Potential problems [Patel, S&P97]:

- If an active attacker, instead of sending g and p in clear, chooses to send g^d and p such that d is a small prime and d|(p-1). Then, $(g^{dy})^{(p-1)/d} = 1 \bmod p$. When the attacker receives the password encrypted $ENC_{pwd}(g^y)$, he tries to decrypt it with different candidate passwords and raises the decrypted number to (p-1)/d. If the result is not 1 then that password is rejected. Since (p-1)/d number out of p-1 number will be dth power residue, hence 1/d numbers on average will be congruent to 1 when raised to (p-1)/d. At each session the possible space of password is reduced to 1/d and the space of valid passwords will be narrowed to 1 at a logarithm rate (typically, logp).
- Avoidance: The success of the attack is due to the fact that g^d is not a generator. To find a generator g it is necessary and sufficient to check that $g^{(p-1)/m} \neq 1 \bmod p$ for all factors m of p-1.

[BPR Eurocrypt2000]



[BPR Eurocrypt2000]

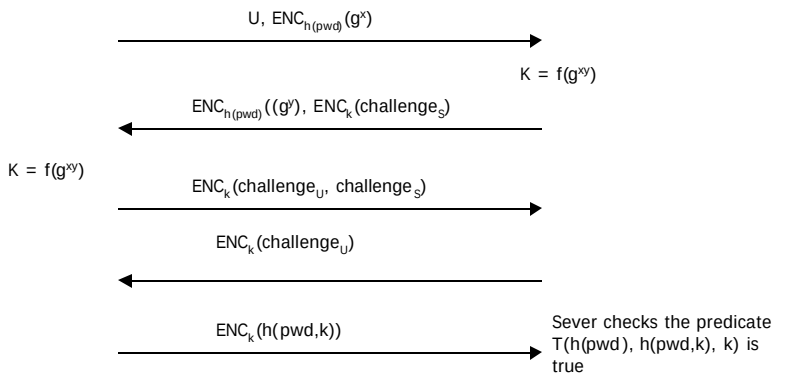
The [BM92] can only be proved secure in this year in ROM and ICM.

Theorem. Let q_{se}, q_{re}, q_{co}, q_{ex}, q_{or} be integers and let $q = q_{se} + q_{re} + q_{co} + q_{ex} + q_{or}$. Let Password be a finite set of size N and assume $(|Ì|)^{1/2}/q \geq N \geq 1$. Let PW be the associated LL-key generator as discussed above, SK be the associated session key space. Assume the weak corruption model. The

$Adv^{fs}_{P,PW,SK}(t,q_{se},q_{re},q_{co},q_{ex},q_{or}) <= q_{se}/N + q_{se} \cdot q_{or}\ Adv^{dh}_{Ì,g}(t',q_{or}) + O(q^2)/|Ì| + O(1)/(|Ì|)^{1/2}$

Where $t' = t + O(q_{se}+q_{or})$.

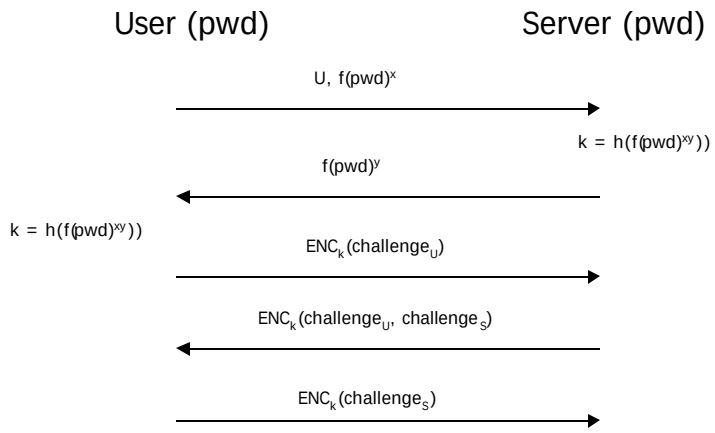
AEKE: [BM, CCS93]



AEKE: [BM CCS93]

- Motivation: An attacker break into a server still needs to impose dictionary guessing before impersonating an authorized user.
- The predicate T(h(pwd), h(pwd,k), k) was defined as following [BM93]:
  - Digital signature-based: h(pwd) is the public key, whereas h(pwd,k) is the signature on k. Therefore, the public key is used for EKE exchange, and the private key is used for AEKE extension.
    - [BM93]: Any number (e.g., pwd or some simple function of it) can be private key.
  - Commutative hash function-based: Let h(pwd) be defined as h_0 (p), a member of a family of commutative one-way hash functions, {h_0, h_1, ...}. After the EKE transfer using h(pwd)=h_0(pwd), both participants generate the hash function h_k(). Finally,$h_0(h_k(pwd))=h_k(h_0(pwd))$.
    - Right now, we do not know any such hash functions.
    - The RSA-based hash functions are subject to attacks. Define $h_k(pwd) = (pwd)^k$ and $h_0(pwd) = (pwd)^l$, then $h_0(h_k(pwd))=h_k(h_0(pwd))$. However, if (l,k)=1, then pwd can be computed using Simmons' attack with probability 61%; If t = gcd(l,k) >1, then $(pwd)^t$ can be found out and it can be used to compute $((pwd)^t)^s$ for any s.

SPEKE: [Jablon, CCR96]



PAK: [BMP, Eurocrypt2000]

- EKE (or AKE) proven secure using random oracle assumption and "ideal block cipher" assumption [BPR00]
- PAK proven secure using random oracle assumption [BMP00]
  - PAK: using $g^x \cdot H(pwd)$ instead of $ENC_{H(pwd)}(g^x)$

PAK (simplified)

pwd

pwd

$W = g^x \cdot H_1(pwd)$

W

$V = g^y$

$Z = (W/H_1(pwd))^y$

$V, H_{2a}(g^{xy}, pwd)$

$Z = V^x = g^{xy}$

$H_{2b}(g^{xy}, pwd)$

$K = H_3(g^{xy}, pwd)$

$K = H_3(g^{xy}, pwd)$

PAK

pwd

Public:
Primes **p,q, p=rq+1**,
generator **g** of order **q**

pwd

$W = g^x \cdot (H_1(A|B|pwd))^r$

"A",W

Verify $W \neq 0 \bmod p$

$V = g^y$

$Z = (W/(H_1(A|B|pwd))^r)^y$

$V, H_{2a}(A|B|W|V|Z|pwd)$

$Z = V^x$

Verify $H_{2a}(A|B|W|V|Z|pwd)$

$H_{2b}(A|B|W|V|Z|pwd)$

Verify $H_{2b}(A|B|W|V|Z|pwd)$

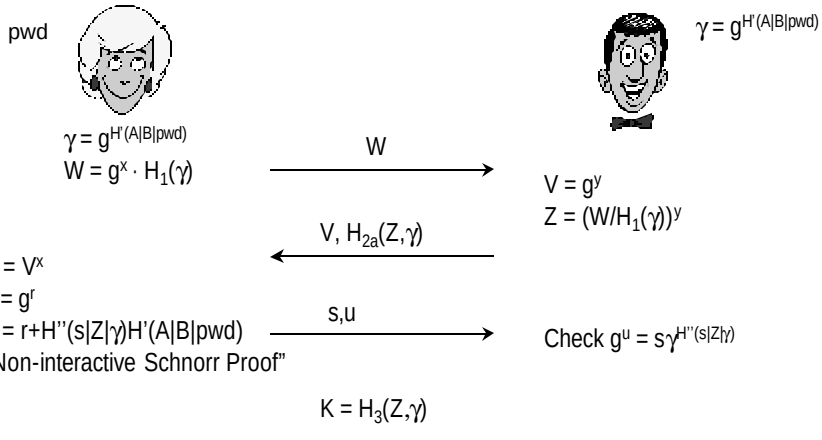
$K = H_3(A|B|W|V|Z|pwd)$

$K = H_3(A|B|W|V|Z|pwd)$

PAK-X

- Resilient to server-compromise
- Alice = Client, Bob = Server
- Bob stores $\gamma = g^{H'(A|B|pwd)}$ instead of pwd
- Extra Proof of Knowledge used to prove Alice knows pwd

PAK-X (simplified)



- Background knowledge
- Previous solutions to authentication: The Trouble
- Being implemented proposals: Server has a cert.
- Weaker assumption proposals: Server has no cert.
- Password protocols and public key cryptography
- Future directions

Is PKC Necessary?

- All the above protocols (and many other not mentioned protocols) used public key cryptography (warning: it doesn't necessarily mean a public-key infrastructure!)
- Is public key cryptography necessary in constructing password-based authentication protocol secure against offline dictionary attack?
- The answer is strongly affirmative.

From PA to SKE

[HK99]: How to construct a a Secure Key Exchange (SKE) protocol from a secure Password Authentication (PA) protocol resistant to offline guessing attack?

Definition [HK99]. A two-party protocol (A, B) is a key-exchange protocol for one bit if at the end of the protocol both A, B output the same bit with respect to public security parameter k. Let $\varepsilon(\cdot)$ be a function and let (A, B) be a key-exchange protocol for one bit. We say that (A, B) is secure up to ε if no feasible eavesdropper E can guess the bit that A, B output with probability better than $\frac{1}{2} + \varepsilon(k)$.

Theorem [HK99]. Any protocol that ensures one-way password authentication up to $\varepsilon(k, l, m)$ can be transformed into a key exchange protocol for one bit, which is secure up to $\varepsilon'(k) = \varepsilon(k, 1, 1)$.

Notation. $\varepsilon(k, l, m)$ means at most m active impersonation attempts, and in which the legitimate user outputs at most l successful sessions.

Construction: From PA to SKE

The construction. We assume a dictionary of size 2. Therefore, a secure password protocol resistant to offline dictionary attack implies that the attacker cannot guess the user's password with ½ plus non-negligible probability.

To exchange 1 bit, each of the parties chooses at random a password from the dictionary and then execute the password protocol with one party playing the role of server and the other playing the role of client.

If the execution succeeds, then their secret bit is set to '0' in the case that the password that they chose was the first password in the dictionary, or to '1' otherwise.

After the expected 2 trials, the agree on 1 bit.

The Reduction

We need to prove the completeness and the soundness.

Completeness. The probability q that the two parties output different bits is negligible.

The event that they output different bits only happens if the passwords A and B pick in the first execution of the password protocol are different, and either this first execution is successful (even the password is different, i.e., we allow a probability error. we denote the probability p) or else the bit chosen in the remains of the protocol is not the same. Thus, we have $q = ½ (p + (1-p)q)$.

Now we need to evaluate the bound of p. Since the password protocol is secure, we consider an attacker who simply guesses a password from the dictionary at random and tries an authentication session with the server using that password. Since the probability for the attack to hit the correct password is ½ , we have:

$½ + \varepsilon \geq$ Pr [attacker succeeds]

$=$ ½ · Pr [attacker succeeds | his guess is correct] + ½ · Pr [attacker succeeds | his guess is incorrect]

$= ½ + ½ \cdot p$

Consequently, $p \leq 2\varepsilon$, and $q < p < 2\varepsilon$ (i.e., negligible).

The Reduction

Soundness. We need to prove that the key exchange protocol is $\varepsilon'(k) = \varepsilon(k, 1, 1)$ secure.

We reduce the security of the key exchange to the security of the password authentication protocol. If there exists an eavesdropper adversary E with advantage of more than ε' in guessing the exchanged bit, then we use it as an oracle to break the password authentication protocol with the same advantage, thus contradicts to the security of the password authentication protocol.

The attacker runs an execution query for the password authentication protocol of dictionary size 2, thus he obtained a genuine transcript τ. Then he simulates the secret key exchange protocol. That is, he repeatedly picks pairs of passwords from the dictionary (which is fresh for every runtime, just as the real key exchange protocol) of size 2. As long as they are different (i.e., the one for A is different from the one for B), then he obtained a transcript (we denote is by τ_1) for this runtime. This process is not stopped until the two passwords for A and B are equal at (say) time i.

Now, we give the transcript $<\tau_1, ..., \tau_{i-1}, \tau>$ to E and we bet the password on the answer of E. Therefore, we succeed with probability more than $\varepsilon'(k) = \varepsilon(k, 1, 1)$.

PKC and Passwords

- Secure secret-key agreement is known to be possible under the assumption that trapdoor functions exist [DH76, GM84].
- Question: Can we base secure secret-key agreement on the existence of one-way permutations only?
- Answer [IL STOC89]: probably no (see argument below)

Impagliazzo & Rudich STOC89

- Motivated by the question: Can we base secure secret-key agreement on the existence of one-way permutations only?
- They depicted a model in which only secret key tools are available. This model is different from the standard model in the following ways:
  - NP-Complete oracle: To eliminate all public key tools from the model, the adversary is given such an oracle to solve the public key related problem (i.e., compute the private key from a public key).
  - Random oracle: To enable non-public key tools in the presence of such a NP-C oracle, all the parties in this model are given access to a random function f, mapping arbitrary-length strings into strings of length k (i.e, the security parameter). Namely, for each string in {0,1}*, f maps it to an independent, uniformly distributed k-bit string. Within such a Random Oracle, various primitives (e.g., collision-resistance hashing, symmetric encryption, etc.) are easy to implement.

Impagliazzo & Rudich STOC89

Theorem [IR89]. There is no key-exchange protocol in the IR-model which is secure up to any $\varepsilon < ½$.

Theorem [HK99]. There is no password authentication protocol in the IR-model which is secure up to any $\varepsilon < ½$.

Why OWP is not enough?

- In the IR-model, only non-public key tools are available.
- Claim. In the standard model, construct a password protocol that can be provably secure based only on the security of symmetric key tools (e.g., symmetric encryption, MAC, collision-resistant hashing) is at least as hard as proving that P≠NP.
  - Assume that we have such a password protocol in the standard model.
  - Since the derived key exchange protocol is secure with only symmetric key tools, it is still secure in the IR-model.
  - However, in IR-model, the existence of any NP-Complete oracle is enough to break any key exchange protocol.
  - Therefore, there are no efficient oracles can solve NP-Complete problems.
  - Consequently, P≠NP.

A Corollary

Theorem [B99]. Secure password authentication protocol exists (in the robust definition sense) if secure key exchange protocol exists.

This is due to the fact that the password authentication protocol needs only a secret key exchange (i.e., we exchange password) and a signature scheme. Moreover, secret key exchange (at least we need OWF thereof) implies the existence of a signature scheme.

Corollary. $\exists$ secure password protocol $\Leftrightarrow \exists$ secure key exchange protocol.

Future Direction

This is only part of the more comprehensive problem we call Authentication and Authenticated Key Exchange (A&AKE). There are still a lot of challenges.

- How can we reach a consistent world?
- Adaptive security ...
- Stronger notions ...