Part III: Protocols

Protocol

- Human protocols — the rules followed in human interactions
  - Example: Asking a question in class
- Networking protocols — rules followed in networked communication systems
  - Examples: HTTP, FTP, etc.
- Security protocol — the (communication) rules followed in a security application
  - Examples: SSL, IPSec, Kerberos, etc.

Protocols

- Protocol flaws can be very ***subtle***
- Several well-known security protocols have significant flaws
  - Including WEP, GSM, and IPSec
- Implementation errors can also occur
  - Recently, IE implementation of SSL
- Not easy to get protocols right…

نقص پروتکل می تواند بسیار ظریف باشد --> یک تغییر کوچیک توی پروتکل باعث میشه ماهیت اون به کلی عوض بشه همین تغییر کوچیک میتونه باعث اسیب پذیری پروتکل هم بشه
نکته: توی پروتکل های امنیتی هم اسیب پذیری و نقص هم داشتیم و حواسمون باید باشه از اخرین نسخه استفاده بکنیم چرا اخرین نسخه؟ چون حداقل اسیب پذیری رو دارند
نکته: بعضی وقتا خود پروتکل مشکلی نداره و کسی که اومده اونو پیاده سازی بکنه باعث شده در پیاده سازی خطایی به وجود بیاد مثل پیاده سازی SSL
نکته: خیلی اسون نیست که یه پروتکل همه چی تموم داشته باشیم
چندین پروتکل امنیتی شناخته شده دارای نقص های قابل توجهی هستند

- شامل GSM، WEP و IPSec

خطاهای پیاده سازی نیز ممکن است رخ دهد

- اخیراً اجرای IE از SSL

درست کردن پروتکل ها آسان نیست...

Ideal Security Protocol

- Must satisfy security requirements
  - Requirements need to be precise
- Efficient
  - Minimize computational requirement
  - Minimize bandwidth usage, delays…
- Robust
  - Works when attacker tries to break it
  - Works if environment changes (slightly)
- Easy to implement, easy to use, flexible…
- Difficult to satisfy all of these!

باید الزامات امنیتی را برآورده کند
- الزامات باید دقیق باشد
  کارآمد
- نیاز محاسباتی را به حداقل برسانید
- به حداقل رساندن استفاده از پهنای باند، تاخیر...
  قدرتمند
- زمانی کار می کند که مهاجم سعی می کند آن را بشکند
- اگر محیط تغییر کند (کمی) کار می کند
  آسان برای پیاده سازی، آسان برای استفاده، انعطاف پذیر...
  ارضای همه اینها سخت است!

اگر سوالی از ما می پرسه ارزیابی ما چیه ما باید به تمام نکات بالا توجه بکنیم:

1- اولا نیازمندی امنیتی ما رو رفع بکنند مثلا این اهداف امنیتی که برای کربروس طرح کردیم بتونه پاسخ بده

2- بحث کارایی پروتکل --> مثلا کم کردن محاسبات ینی توجه میکردیم که کلید عمومی محسابات سنگینی نیاز داره پس حداقل محاسبات رو نیاز داره

3- مقاومت یک پروتکل یا robust: ینی حتی زمانی که ما تحت حمله هستیم و مهاجم داره سعی میکنه که ما رو شکست بده توی سیستم ما باید بازم کار بکنیم و این خیلی مهمه که اون ویژگی avalibility رو سعی بکنیم حفظ بکنیم پس:

1- پس تحت حملات هم باید کار بکنیم و کاربر نرمال ما از سرویس ما محروم نشه توی چنین مواردی باید یک سرویس پشتیبان داشته باشیم و وقتی سرویس اصلی مورد هجوم واقع شد سرویس پشتیبان باید سرویس بده که اون دسترسی رو از دست ندیم

پس تحت حمله سیستم و شبکه هم باید کار بکنه

2- تغییرات محیطی هم باعث نشه که پروتکل ما کار نکن و وقتی پروتکل داریم طراحی میکنیم به این هم باید جواب بدیم پس هر چقدر پروتکل ما بتونه خودشو طبیق بده با محیط های مختلف این یک مزیت واسه پروتکل ما حساب میشه

4- پیاده سازی پروتکل ساده باشه و پیچیده نباشه چون اگر پیچیده باشه توی اون پیاده سازیش اون خطاهه احتمالا اتفاق بیوفته

5- کاربر پسند باشه و منعطف باشه

رسیدن همزمان به همه این ویژگی ها خیلی سخته و بعضی وقتا از بین اینا اومده یکی رو فدای یکی دیگه کرده نکته: ولی اگر ارزیابی امنیتی ما چیه ما فقط روی خط اول همین جا بحث میکنیم

Chapter 9: Simple Security Protocols

"I quite agree with you," said the Duchess; "and the moral of that is—
'Be what you would seem to be' —or
if you'd like it put more simply—'Never imagine yourself not to be
otherwise than what it might appear to others that what you were
or might have been was not otherwise than what you
had been would have appeared to them to be otherwise.' "
— Lewis Carroll, *Alice in Wonderland*

Seek simplicity, and distrust it.
— Alfred North Whitehead

Secure Entry to NSA

1. Insert badge into reader
2. Enter PIN
3. Correct PIN?

   Yes? Enter

   No? Get shot by security guard

مثلا پروتکل ورود امن ما به یک جایی مثل اژانس امنیت ملی به این صورته که یک badge به ما دادن و در این حالت ما یک pin رو وارد میکنیم و مسئله ای که هست این که پین رو درست وارد کردیم یا نه

ATM Machine Protocol

1. Insert ATM card
2. Enter PIN
3. Correct PIN?

   Yes? Conduct your transaction(s)

   No? Machine (eventually) eats card

کارت های بانکی
وارد کردن کارت
رمز کارت
اگر رمز درست بود می تونیم تراکنش رو انجام بدیم و اگر درست وارد نکرده باشیم کارت رو میخوره

Identify Friend or Foe (IFF)

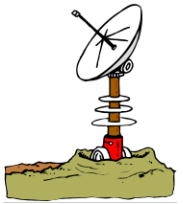
Russian
MIG

Angola

SAAF
Impala
K

2. E(N,K)

1. N



Namibia
K

مسئله:
یک سیستم مهمی که همه جا استفاده میشه سیستم IFF است
و این یک سیستم تشخیصی است و باید تشخیص بده اونچه که داره در اسمان پرواز میکنه یک دشمنه یا نه
یه همچین چیزی توی امتحان می بینیم ولی نه این
کاری که می کرد اون بود که سامانه رادار یک N به عنوان چالش واسه اون چیزی که توی اسمان بود ارسال میکرد و جنگده بخاطر این که قبلا با اون سامانه یک کلید مشترک داشتند می اومد اون چالش رو با اون کلید رمز می کرد و برمی گردوند و سامانه می فهمید این یک دوسته

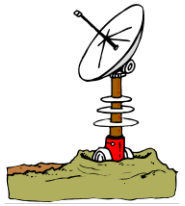
MIG in the Middle

SAAF
Impala
K

3. N

4. E(N,K)

2. N



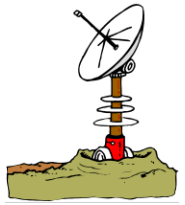
Angola

5. E(N,K)

Russian
MiG

6. E(N,K)

1. N



Namibia
K

حمله که داریم به اسم MIG in the middle:
اگر جنگده SAAF در محدوده پروازی angola وجود نداشت به جنگده mig می گفتن تو حق نداری وارد محدود پروازی namibia بشی
ولی اگر وجود داشت ازروی شکل ببین می فهمی
اینجا هیچ انسانی دخیل نیست
نکته:
اولا جنگده mig باید نزدیک باشه به سامانه angola تا بتونه سیگنال رو بفرسته و SAAF هم باید به رادار angola نزدیک باشه پس این رادار باید به هردوی این جنگنده ها نزدیک باشه
پنجره زمانی که میتونه اینجا انجام بشه هم مهمه ولی اگر این زمان رو هم خیلی کم بگیریم ممکنه دوست خودمون هم بزنیم واسه رادار namibia است
ما مدت زمانی که بگیم هنوز جواب رو نگرفتیم و اگر E(N,K گرفتیم توی این بازه زمانی معتبر است و تعیین این زمان یک نکته مهمی است و ما باید توی چنین سیستم هایی بهش توجه بکنیم و اگر زمان رو خیلی کوتاه بگیریم ممکنه دوست خودمون هم نشناسیم و اگر خیلی زیاد بگیریم ممکنه چنین حمله هایی رو داشته باشیم

Authentication Protocols

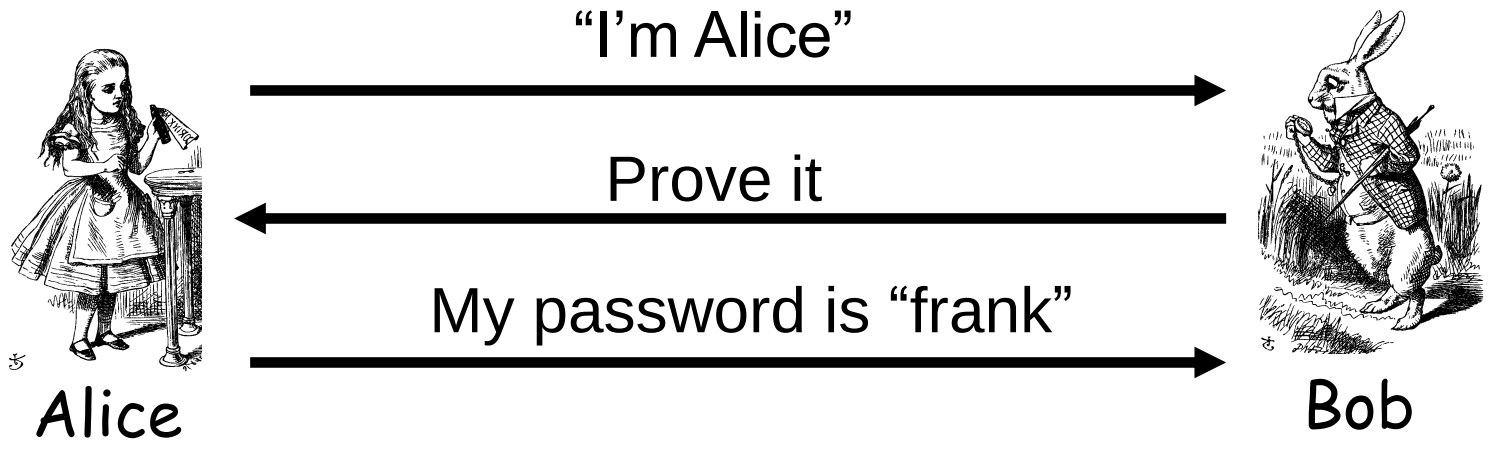
Authentication

- Alice must prove her identity to Bob
  - Alice and Bob can be humans or **computers**
- May also require Bob to prove he's Bob (mutual authentication)
- Probably need to establish a **session key**
- May have other requirements, such as
  - Public keys, symmetric keys, hash functions, …
  - Anonymity, plausible deniability, perfect forward secrecy, etc.

Authentication

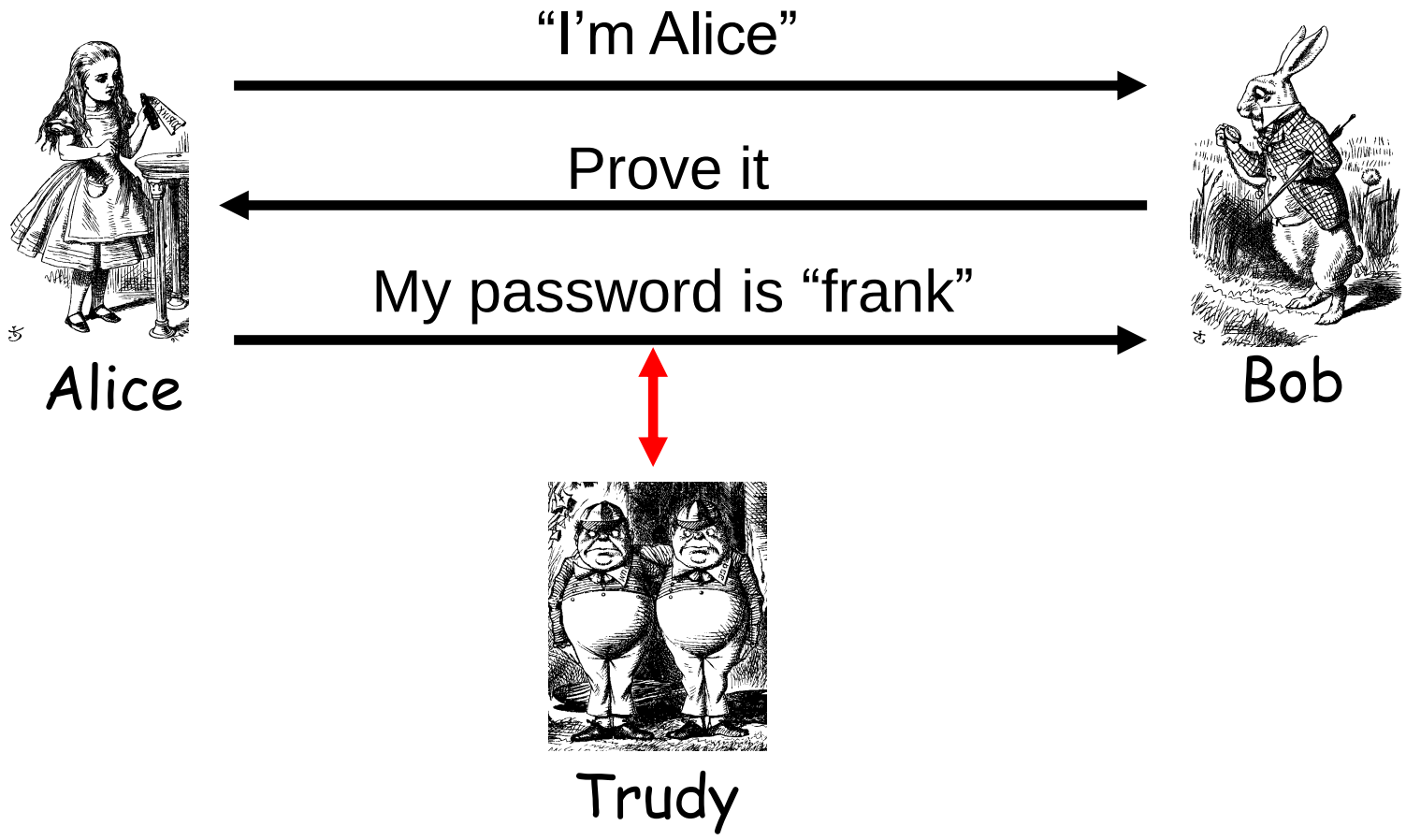
- Authentication on a stand-alone computer is relatively simple
  - For example, hash a password with a salt
  - "Secure path," attacks on authentication software, keystroke logging, etc., can be issues
- Authentication over a network is challenging
  - Attacker can passively observe messages
  - Attacker can replay messages
  - Active attacks possible (insert, delete, change)

Simple Authentication

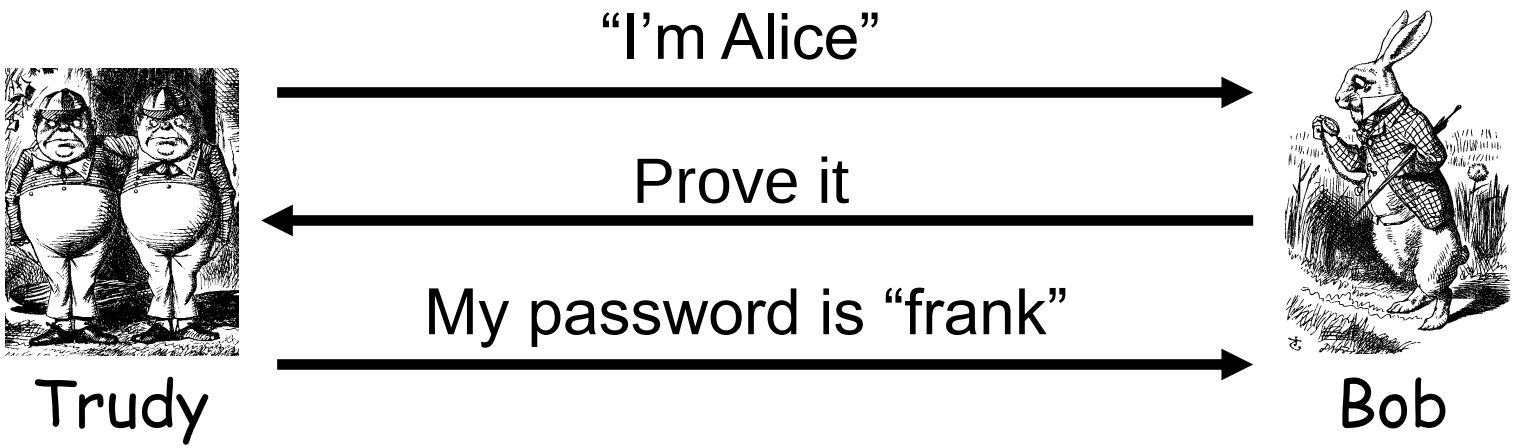


- Simple and may be OK for standalone system
- But highly insecure for networked system
  - Subject to a **replay** attack (next 2 slides)
  - Also, Bob must know Alice's password

Authentication Attack



Authentication Attack



- This is an example of a **replay** attack
- How can we prevent a replay?

Simple Authentication

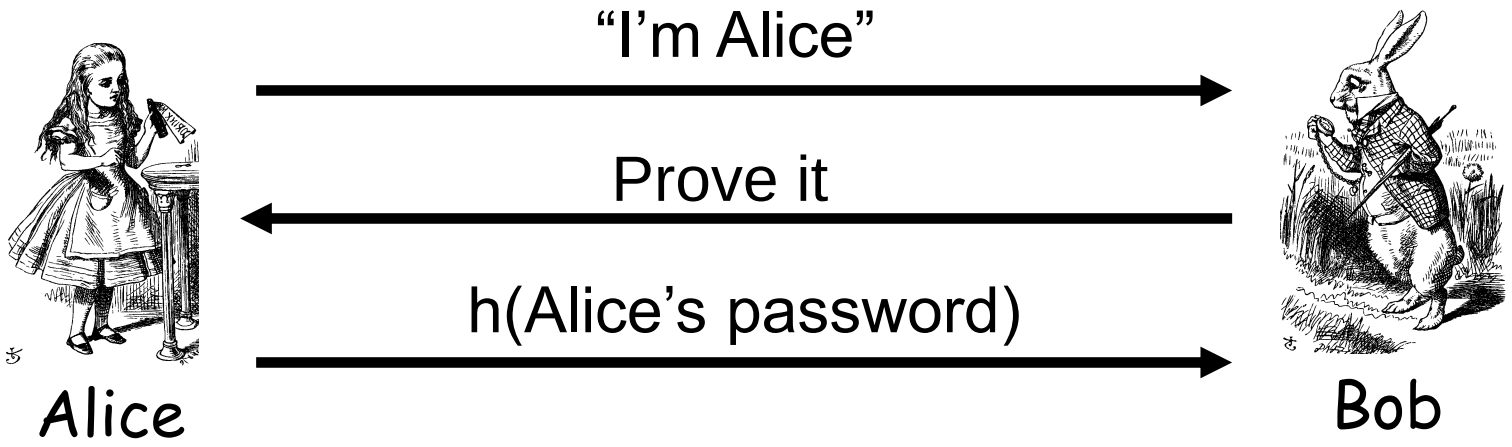
Alice

I'm Alice, my password is "frank"

Bob

- More efficient, but…
- … same problem as previous version

Better Authentication



- This approach hides Alice's password
  - From both Bob and Trudy
- But still subject to replay attack

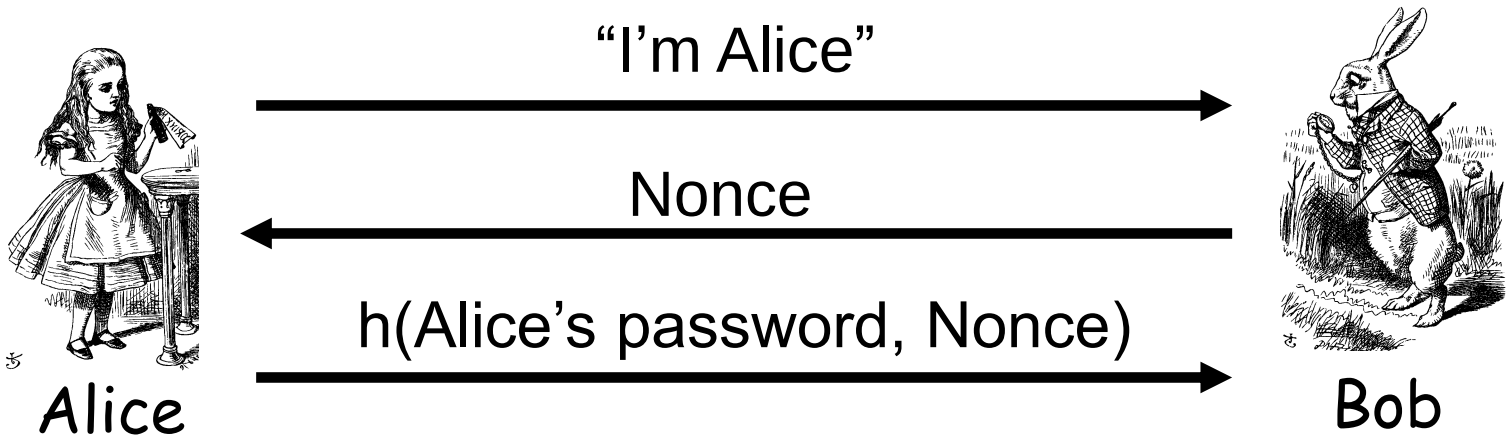
Challenge-Response

- To prevent replay, use ***challenge-response***
  - Goal is to ensure "freshness"
- Suppose Bob wants to authenticate Alice
  - ***Challenge*** sent from Bob to Alice
- Challenge is chosen so that…
  - Replay is not possible
  - Only Alice can provide the correct ***response***
  - Bob can verify the response

Nonce

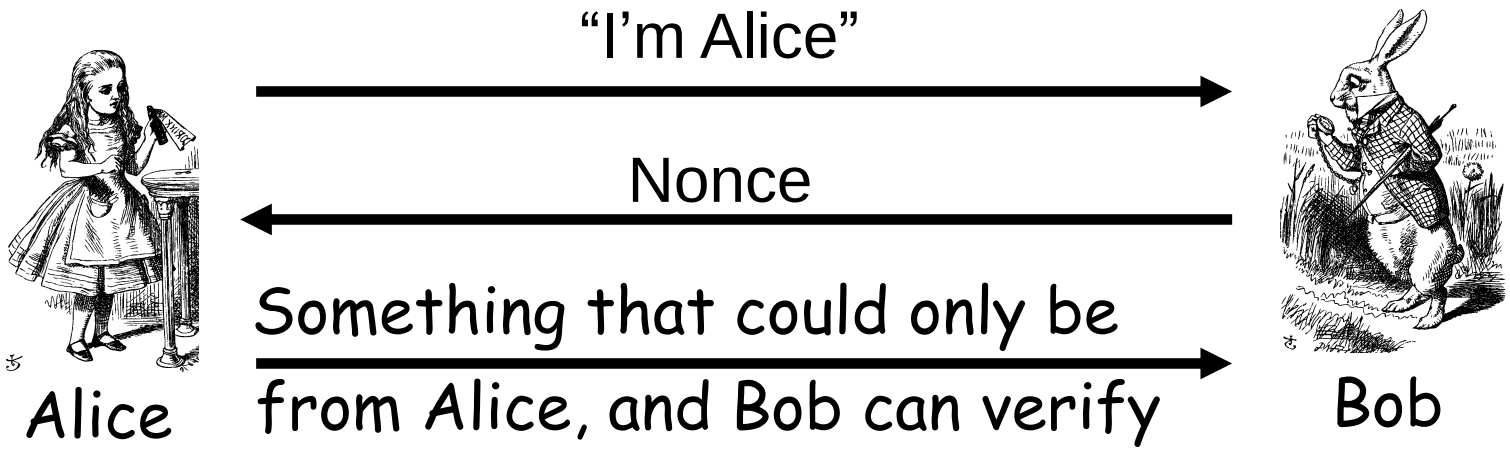
- To ensure freshness, can employ a **nonce**
  - Nonce == **n**umber used **once**
- What to use for nonces?
  - That is, what is the challenge?
- What should Alice do with the nonce?
  - That is, how to compute the response?
- How can Bob verify the response?
- Should we use passwords or keys?

Challenge-Response



- Nonce is the **challenge**
- The hash is the **response**
- Nonce prevents replay (ensures freshness)
- Password is something Alice knows
- Note: Bob must know Alice's pwd to verify

Generic Challenge-Response



- In practice, how to achieve this?
- Hashed password works, but…
- …encryption is much better here (why?)

Symmetric Key Notation

- Encrypt plaintext P with key K

  $C = E(P,K)$

- Decrypt ciphertext C with key K

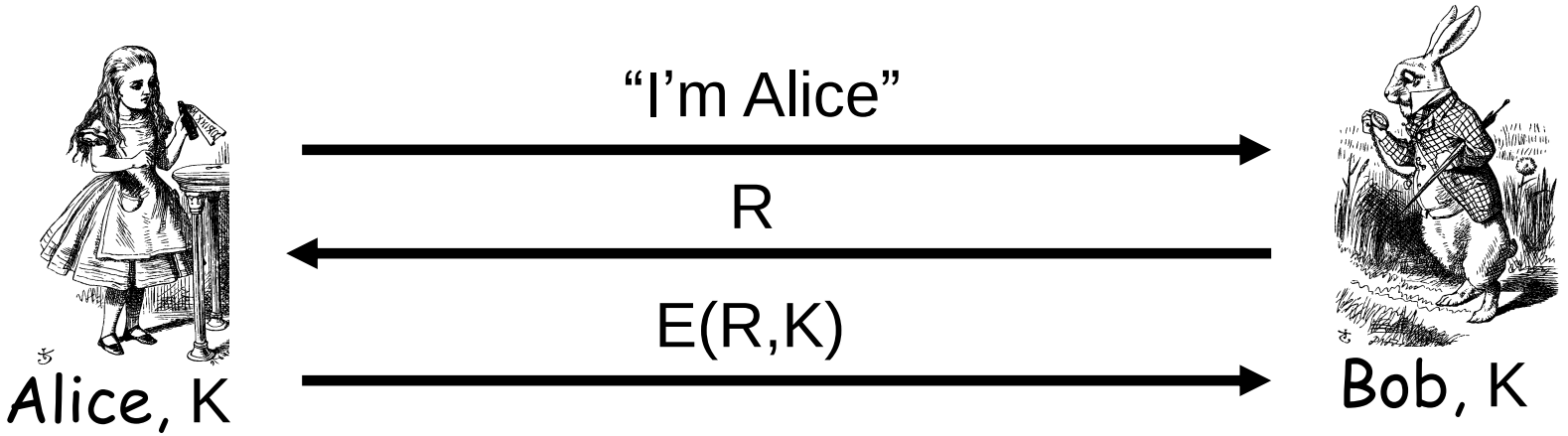
  $P = D(C,K)$

- Here, we are concerned with attacks on protocols, ***not*** attacks on cryptography
  - So, we assume crypto algorithms are secure

Authentication: Symmetric Key

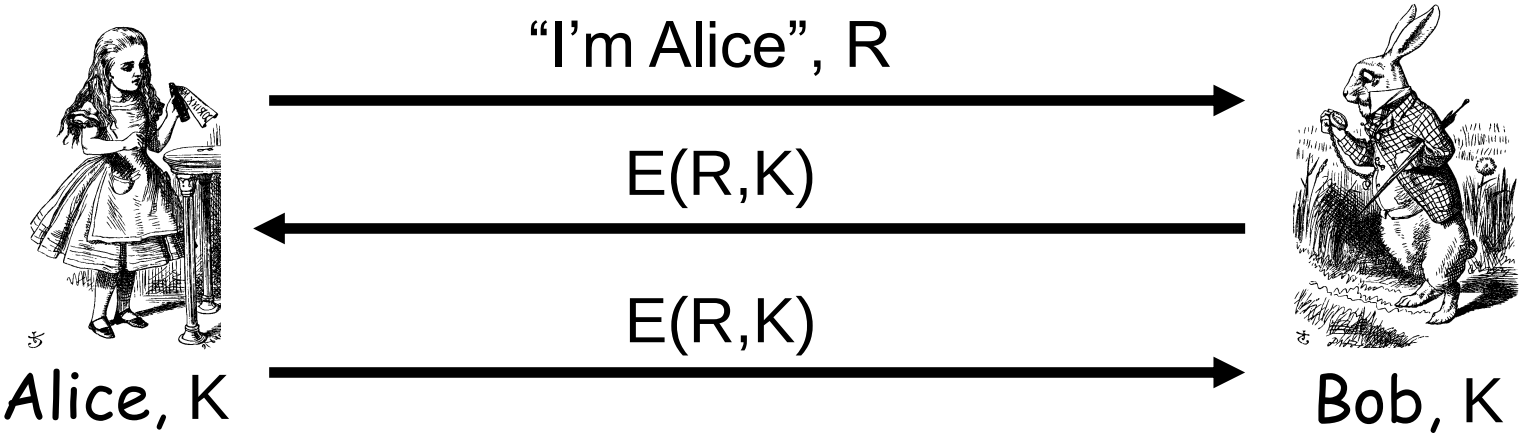
- Alice and Bob share symmetric key K
- Key K known only to Alice and Bob
- Authenticate by proving knowledge of shared symmetric key
- How to accomplish this?
  - Cannot reveal key, must not allow replay (or other) attack, must be verifiable, …

Authenticate Alice Using Symmetric Key



- Secure method for Bob to authenticate Alice
- But, Alice does not authenticate Bob
- So, can we achieve mutual authentication?

Mutual Authentication?

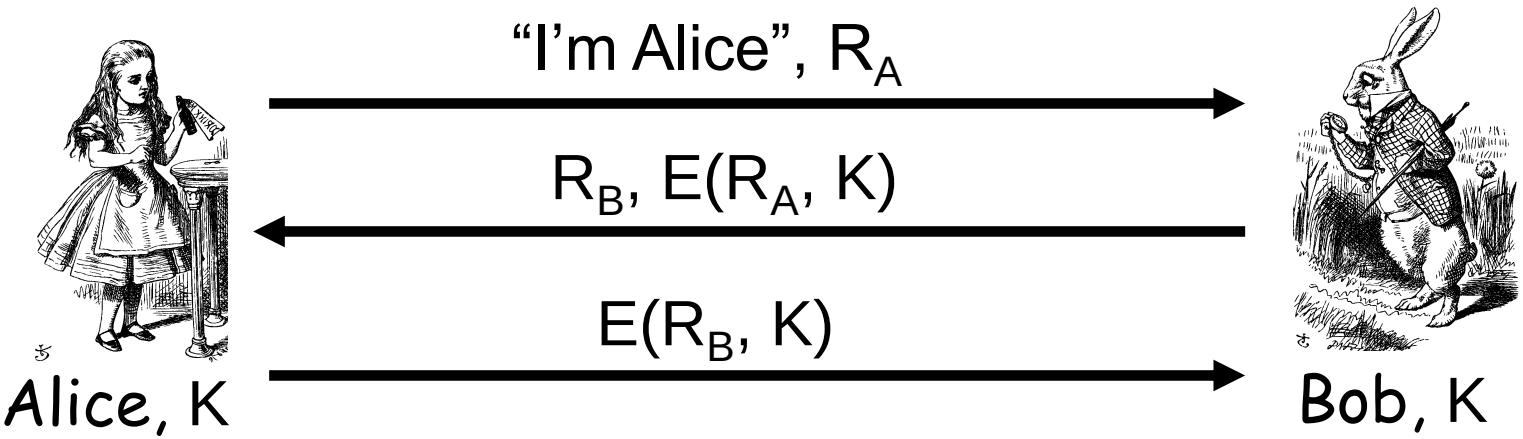


- What's wrong with this picture?
- "Alice" could be Trudy (or anybody else)!

Mutual Authentication

- Since we have a secure one-way authentication protocol…
- The obvious thing to do is to use the protocol twice
  - Once for Bob to authenticate Alice
  - Once for Alice to authenticate Bob
- This has got to work…

Mutual Authentication



- This provides mutual authentication…
- …or does it? Subject to **reflection** attack
  - Next slide

Mutual Authentication Attack

Trudy

1. "I'm Alice", R_A

2. R_B, $E(R_A, K)$

5. $E(R_B, K)$

Bob, K

Trudy

3. "I'm Alice", R_B

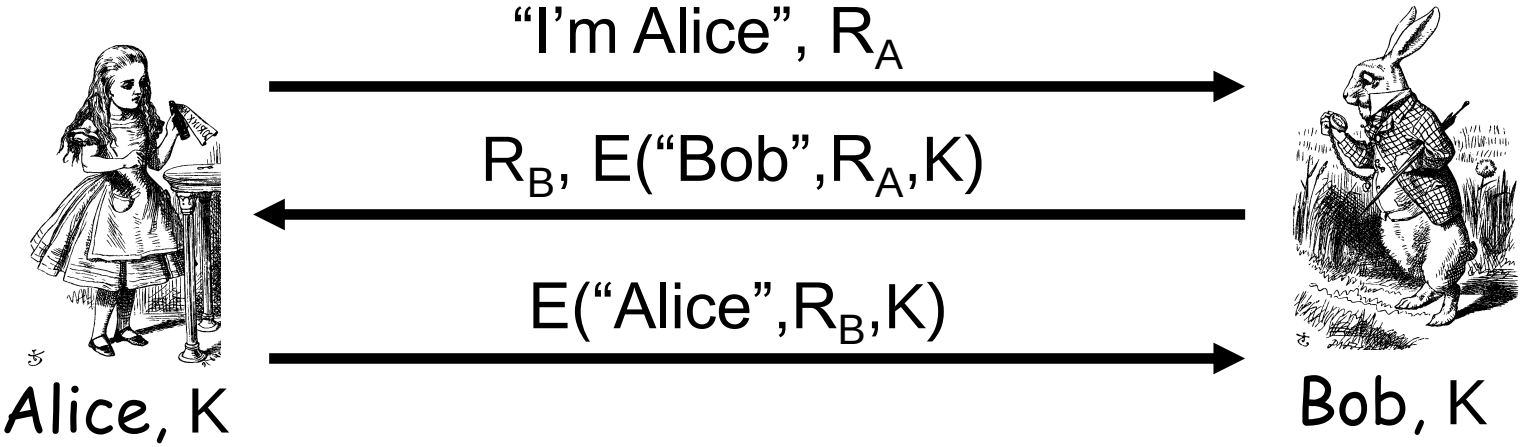
4. R_C, $E(R_B, K)$

Bob, K

Mutual Authentication

- Our one-way authentication protocol is **not** secure for mutual authentication
  - Protocols are subtle!
  - In this case, "obvious" solution is not secure
- Also, if assumptions or environment change, protocol may not be secure
  - This is a common source of security failure
  - For example, Internet protocols

Symmetric Key Mutual Authentication

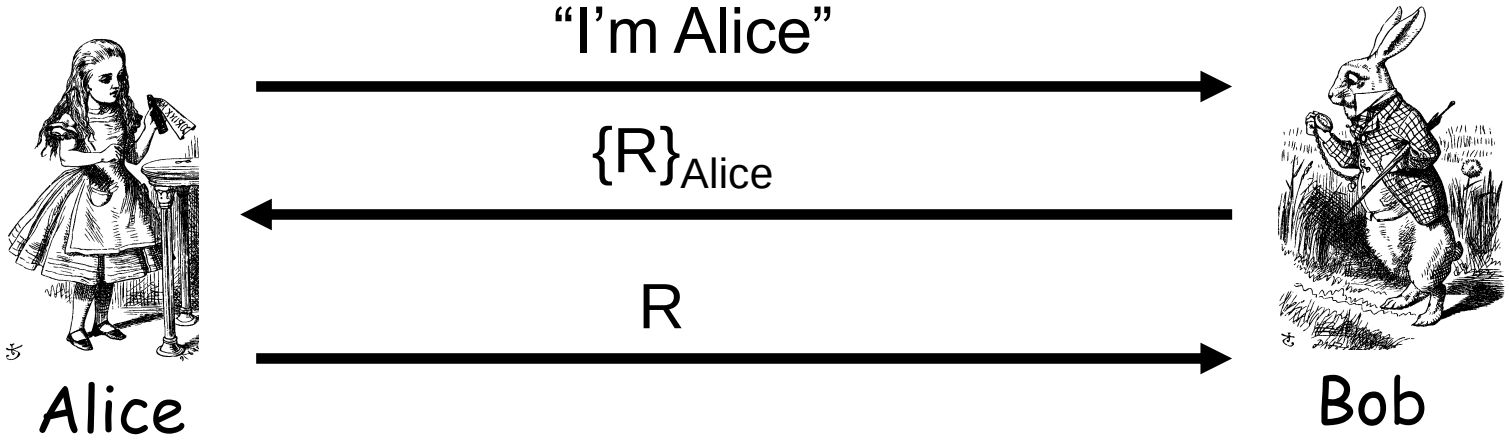


- Do these "insignificant" changes help?
- Yes!

Public Key Notation

- Encrypt M with Alice's public key: $\{M\}_{Alice}$
- Sign M with Alice's private key: $[M]_{Alice}$
- Then
  - $[\{M\}_{Alice}]_{Alice} = M$
  - $\{[M]_{Alice}\}_{Alice} = M$
- **Anybody** can use Alice's **public key**
- Only **Alice** can use her **private key**

Public Key Authentication



- Is this secure?
- Trudy can get Alice to decrypt anything!

  Prevent this by having two key pairs

Public Key Authentication

- Is this secure?
- Trudy can get Alice to sign anything!
  - Same a previous — should have two key pairs

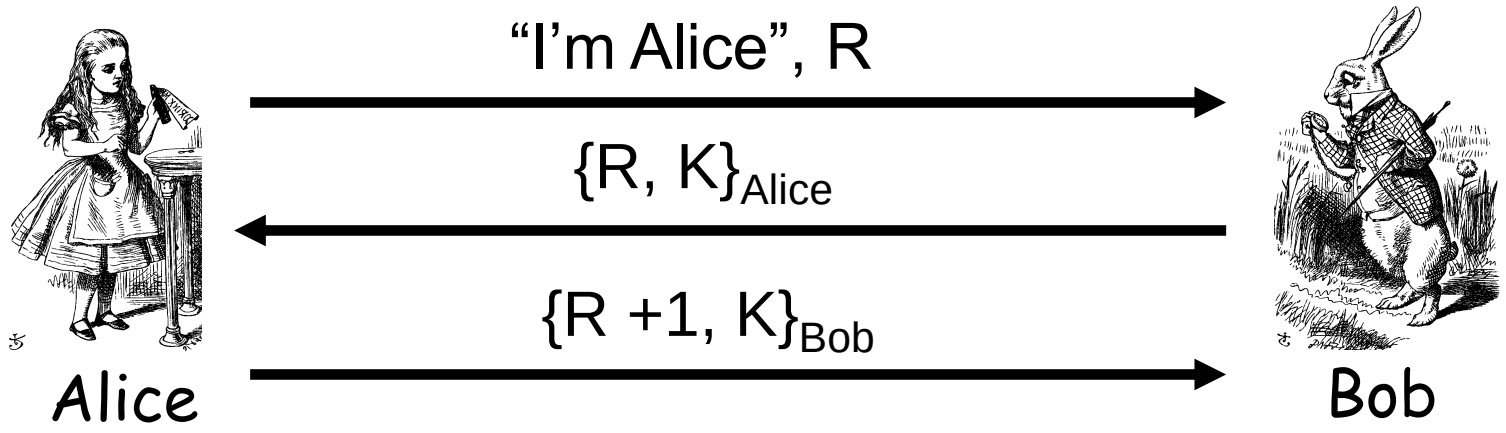
Public Keys

- Generally, a bad idea to use the same key pair for encryption and signing
- Instead, should have…
  - …one key pair for encryption/decryption and signing/verifying signatures…
  - …and a different key pair for authentication

Session Key

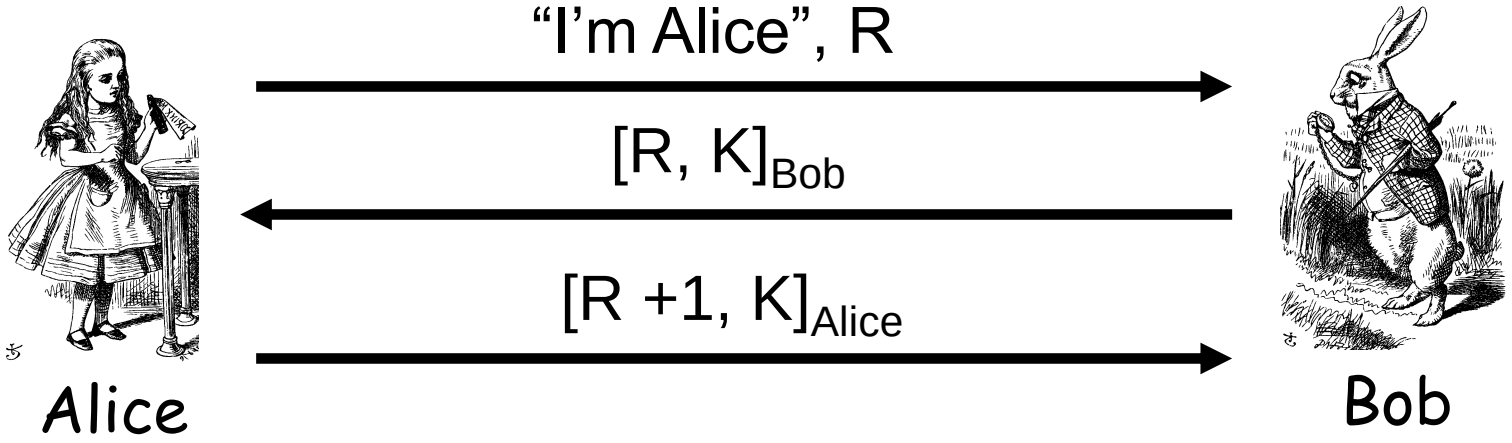
- Usually, a **session key** is required
  - A symmetric key for current session
  - Used for confidentiality and/or integrity
- How to authenticate ***and*** establish a session key (i.e., shared symmetric key)?
  - When authentication completed, Alice and Bob share a session key
  - Trudy cannot break the authentication…
  - …***and*** Trudy cannot determine the session key

Authentication & Session Key



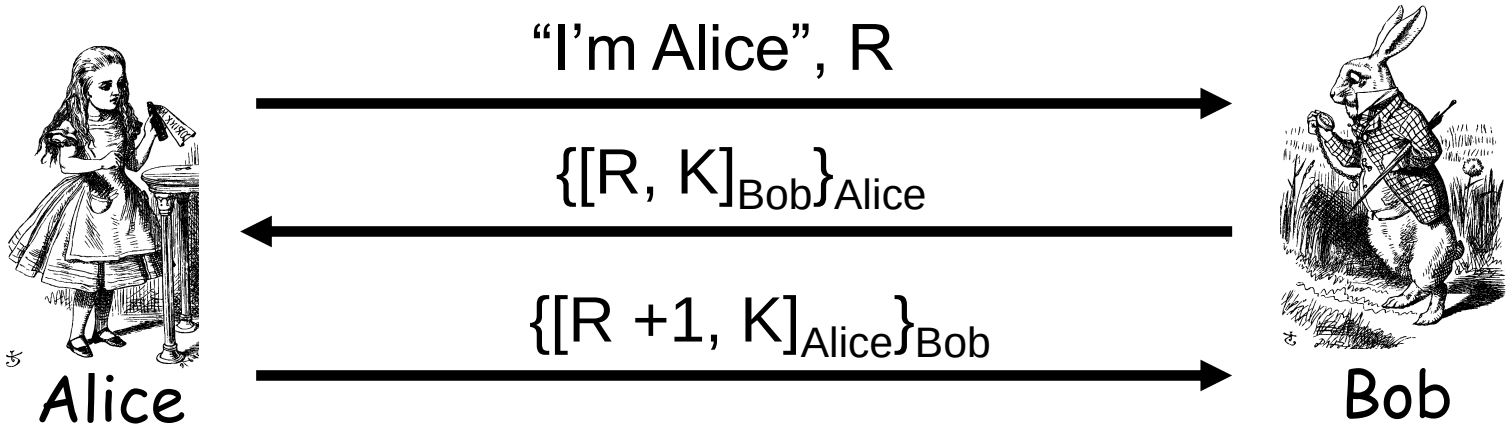
- Is this secure?
  - Alice is authenticated and session key is secure
  - Alice's "nonce", R, useless to authenticate Bob
  - The key K is acting as Bob's nonce to Alice
- No mutual authentication

Public Key Authentication and Session Key



- Is this secure?
  - Mutual authentication (good), but…
  - … session key is not protected (very bad)

Public Key Authentication and Session Key



- Is this secure?
- No! It's subject to subtle MiM attack
  - See the next slide…

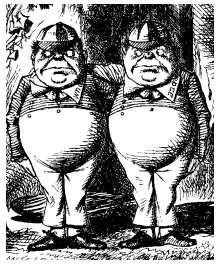
Public Key Authentication and Session Key

Alice

1. "I'm Alice", R

4. $\{[R, K]_{Bob}\}_{Alice}$

5. $\{[R+1, K]_{Alice}\}_{Bob}$



Trudy

2. "I'm Trudy", R

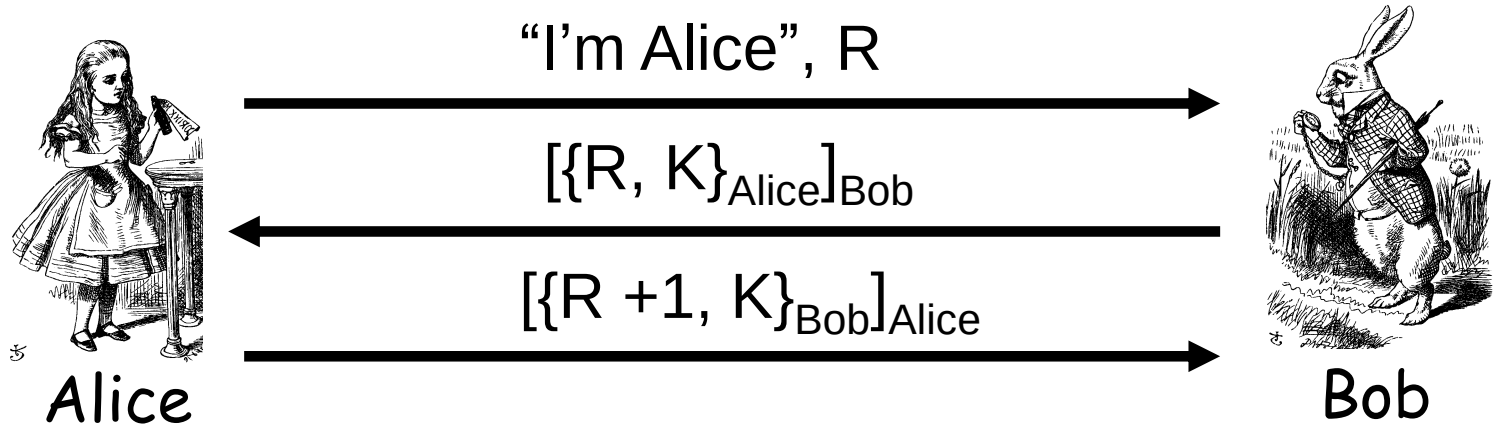
3. $\{[R, K]_{Bob}\}_{Trudy}$

6. time out

Bob

- Trudy can get $[R, K]_{Bob}$ and K from 3.
- Alice uses this same key K
- And Alice thinks she's talking to Bob

Public Key Authentication and Session Key

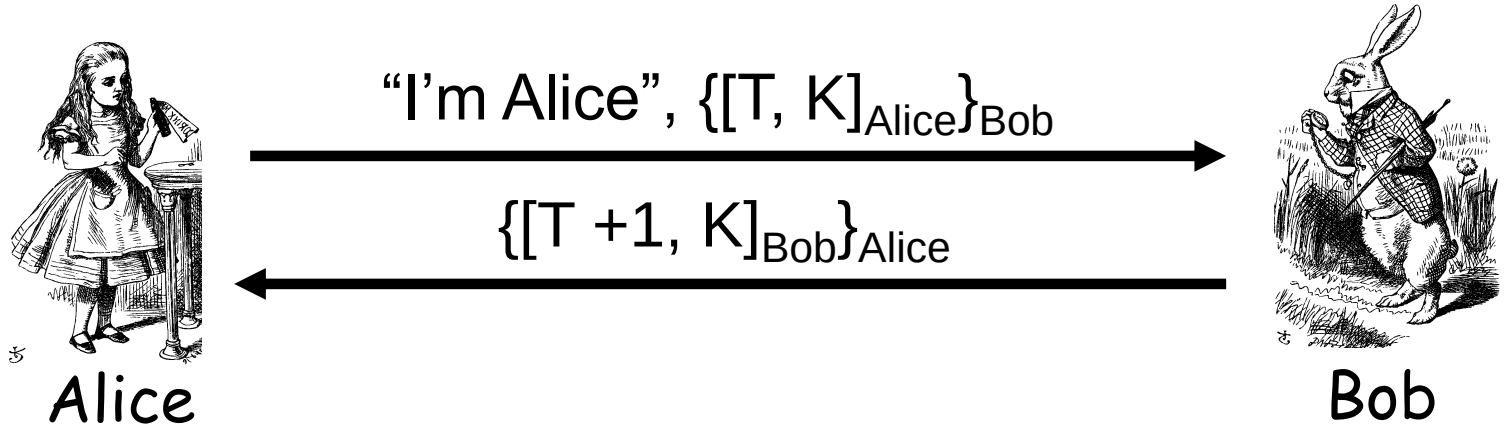


- Is this secure?
- Seems to be OK
  - Anyone can see $\{R, K\}_{Alice}$ and $\{R+1, K\}_{Bob}$

Timestamps

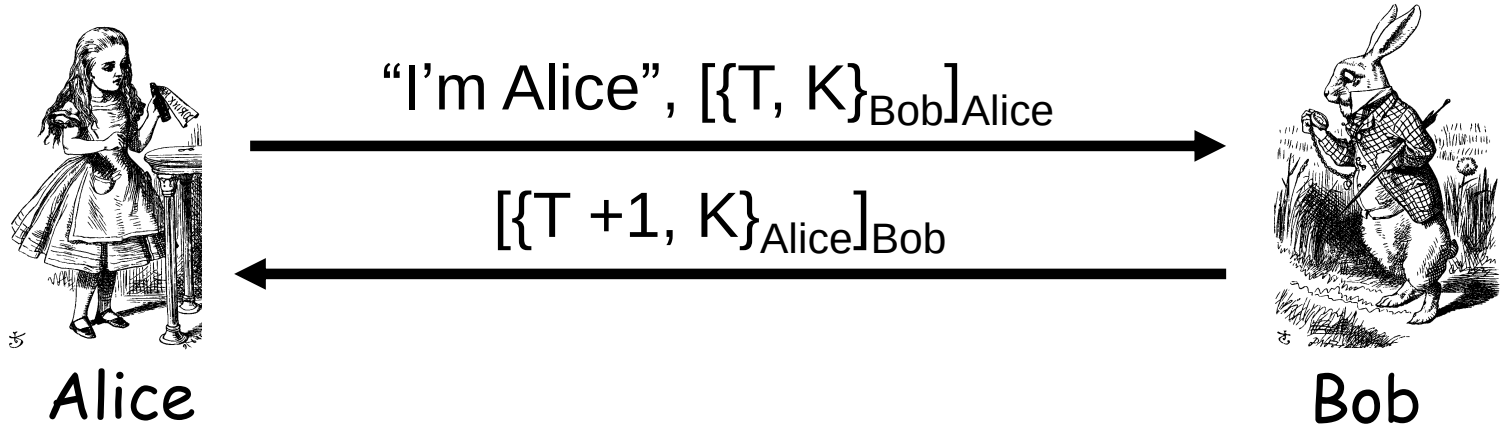
- A timestamp T is derived from current time
- Timestamps can be used to prevent replay
  - Used in Kerberos, for example
- Timestamps reduce number of msgs (good)
  - A challenge that both sides know in advance
- "Time" is a security-critical parameter (bad)
  - Clocks not same and/or network delays, so must allow for **clock skew** — creates risk of replay
  - How much clock skew is enough?

Public Key Authentication with Timestamp T



- Secure mutual authentication?
- Session key secure?
- Seems to be OK

Public Key Authentication with Timestamp T



- Secure authentication and session key?
- Trudy can use Alice's public key to find $\{T, K\}_{Bob}$ and then…

Public Key Authentication with Timestamp T

Trudy

"I'm Trudy", $[\{T, K\}_{Bob}]_{Trudy}$

$[\{T+1, K\}_{Trudy}]_{Bob}$

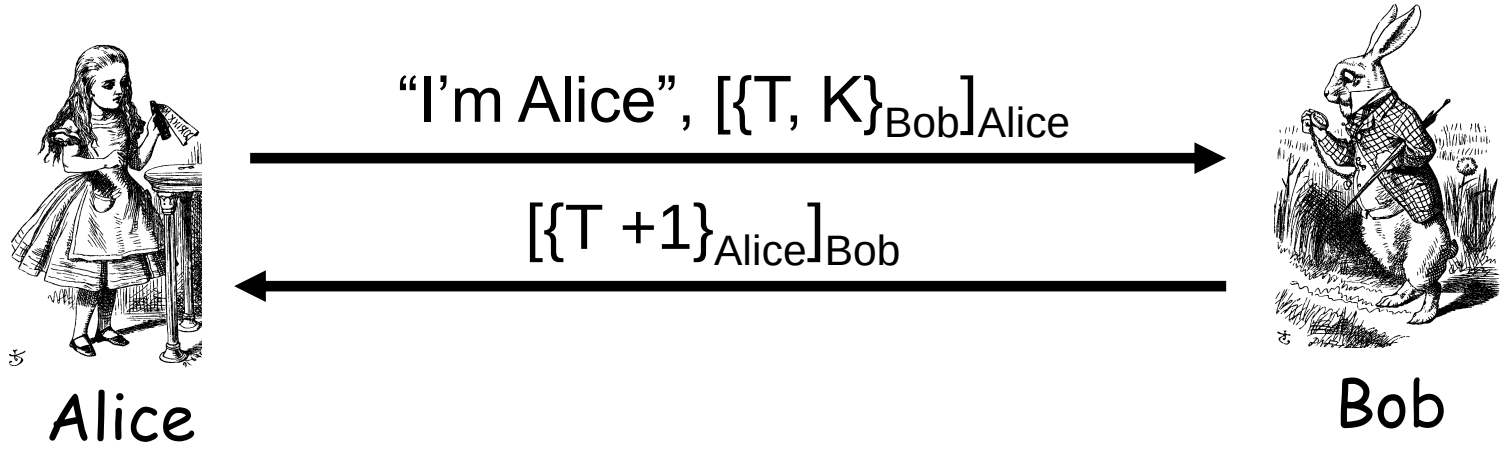
Bob

- Trudy obtains Alice-Bob session key K
- **Note:** Trudy must act within clock skew

Public Key Authentication

- Sign and encrypt with nonce…
  - **Insecure**
- Encrypt and sign with nonce…
  - **Secure**
- Sign and encrypt with timestamp…
  - **Secure**
- Encrypt and sign with timestamp…
  - **Insecure**
- Protocols can be subtle!

Public Key Authentication with Timestamp T



- Is this "encrypt and sign" secure?
  - Yes, seems to be OK
- Does "sign and encrypt" also work here?

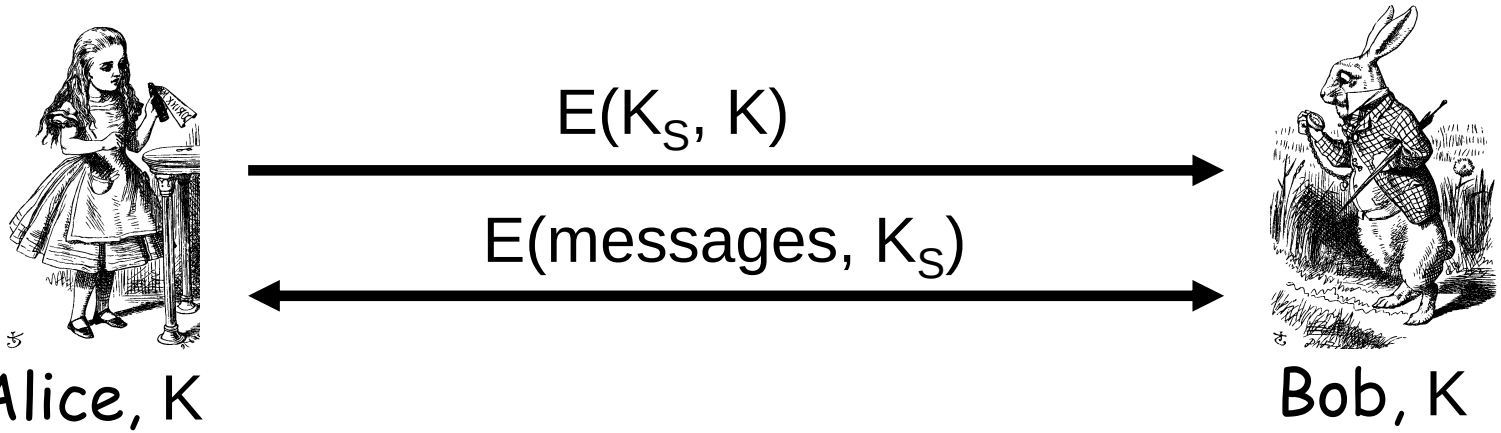
Perfect Forward Secrecy

- Consider this "issue"…
  - Alice encrypts message with shared key K and sends ciphertext to Bob
  - Trudy records ciphertext and later attacks Alice's (or Bob's) computer to recover K
  - Then Trudy decrypts recorded messages
- **Perfect forward secrecy (PFS):** Trudy cannot later decrypt recorded ciphertext
  - Even if Trudy gets key K or other secret(s)
- Is PFS possible?

Perfect Forward Secrecy

- Suppose Alice and Bob share key K
- For perfect forward secrecy, Alice and Bob cannot use K to encrypt
- Instead they must use a session key K_S and forget it after it's used
- Can Alice and Bob agree on session key K_S in a way that provides PFS?

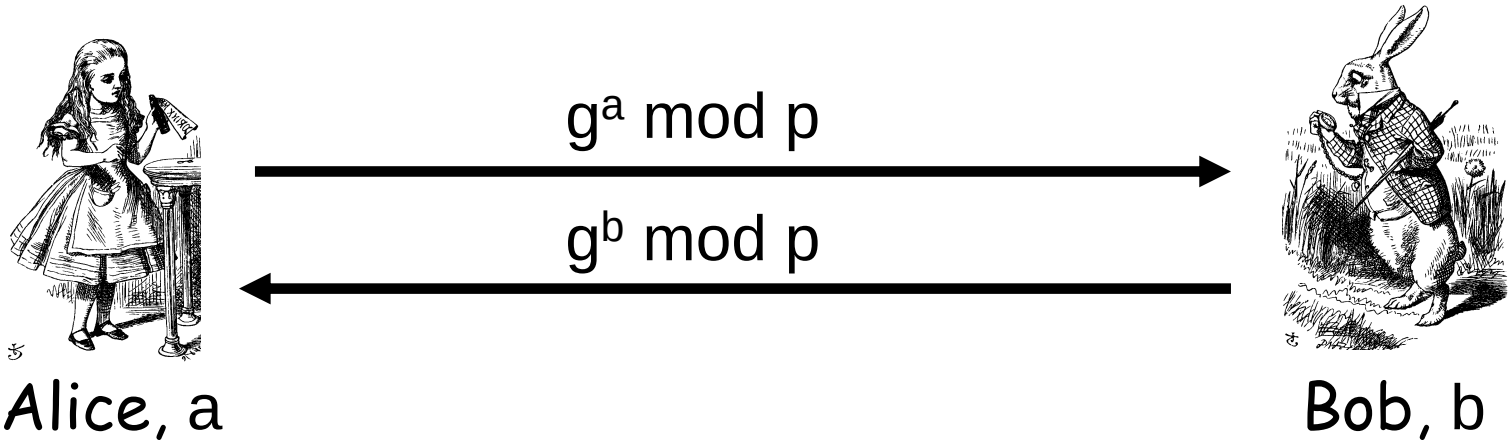
Naïve Session Key Protocol



- Trudy could record $E(K_S, K)$
- If Trudy later *gets* K then she can get K_S
  - Then Trudy can decrypt recorded messages
- ***No*** perfect forward secrecy in this case

Perfect Forward Secrecy

- We can use **Diffie-Hellman** for PFS
- Recall: public g and p



- But Diffie-Hellman is subject to MiM
- How to get PFS and prevent MiM?

Perfect Forward Secrecy

$E(g^a \bmod p, K)$

$E(g^b \bmod p, K)$

Alice: K, a

Bob: K, b

- Session key $K_S = g^{ab} \bmod p$
- Alice **forgets** a, Bob **forgets** b
- This is known as **Ephemeral Diffie-Hellman**
- Neither Alice nor Bob can later recover K_S
- Are there other ways to achieve PFS?

Mutual Authentication, Session Key and PFS

Alice

"I'm Alice", R_A

R_B, $[R_A, g^b \bmod p]_{Bob}$

$[R_B, g^a \bmod p]_{Alice}$

Bob

- Session key is $K = g^{ab} \bmod p$
- Alice forgets a and Bob forgets b
- If Trudy later gets Bob's and Alice's secrets, she cannot recover session key K

Authentication and TCP

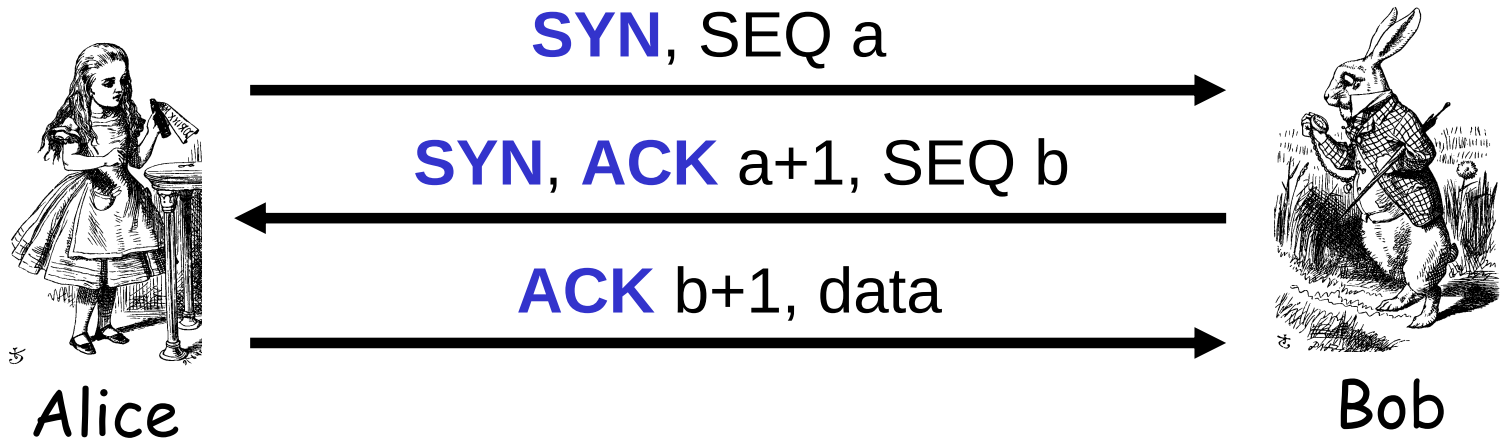
TCP-based Authentication

- TCP not intended for use as an authentication protocol
- But IP address in TCP connection may be (mis)used for authentication
- Also, one mode of IPSec relies on IP address for authentication

TCP برای استفاده به عنوان یک پروتکل احراز هویت/اصالت در نظر گرفته نشده است
اما آدرس IP در اتصال TCP ممکن است برای احراز هویت (سو) استفاده شود
همچنین، یک حالت IPSec برای احراز هویت به آدرس IP متکی است

TCP به عنوان پروتکلی به عنوان احراز اصالت ارائه نشده است پس چون با هدف امنیتی طراحی نشده چنین احراز اصالتی ایمن نخواهد بود
اینجا ما IP ادرس رو داریم و ازش به عنوان ID کاربر استفاده بکنیم و با استفاده از ip ادرس کاربر اونو احراز اصالت هم بکنیم
پروتکل IPSec که با هدف امنیتی طراحی شده توی یکی از حالاتش میاد این کارو هم میکنه ینی کارو بالا رو میکنه و میخوایم ببینیم چه اشکالی داره همچین چیزی...

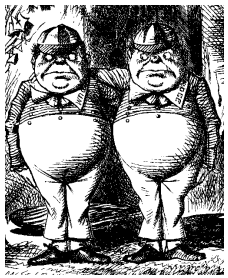
TCP 3-way Handshake



- Initial sequence numbers: SEQ a and SEQ b
  - Supposed to be selected at random
- If not, might have problems…

وقتی الیس و باب بخوان یک ارتباط TCP داشته باشند:
الیس یک پیامی به اسم پیام SYN واسه باب ارسال میکنه و داخلش یک sequence number هم داخلش قرار میده و باب این sequence number یکی اضافه میکنه و ACK اش میکنه و خودش هم یک sequence number می فرسته و بعد الیس این مقدار b رو یکی زیاد میکنه و دیتا هم می فرسته
باب میخواد به همین اتکا بکنه و صرفا با توجه ip ادرسی که الیس داره بگه این ادرس ip به الیس متعلق است و اگر از این ادرس واسه من درخواست اومد حله و دیگه احراز اصالت بشه ینی میگه اگه این پیامو واسه الیس فرستادم و ACK دریافتم از الیس می فهمیم این خود الیس چون میدونیم الیس یک ip خاصی داره و توی شبکه تغییر نمیکنه فرض میکنیم و چون باب این پیامو واسه اون ادرس خاص فرستاده فقط کسی متونه جواب بده که این b رو دیده باشه فرض ما این است که ترودی پیام باب رو نمی بینه با فرض این که ترودی b رو نمی بینه میخوایم ببینیم پروتکل امن هست یا نه؟ زمانی که ترودی اون رو نمی بینه نکته ای که به وجود میاد این که ایا نمی تونه اونو حدس بزنه یا نه؟
ص بعدی...

TCP Authentication Attack



Trudy

1. SYN, SEQ = t (as Trudy)

2. SYN, ACK = t+1, SEQ = b_1

⋮

3. SYN, SEQ = t (as Alice)

5. ACK = b_2+1, data

Bob

4. SYN, ACK = t+1, SEQ = b_2

5.

5.

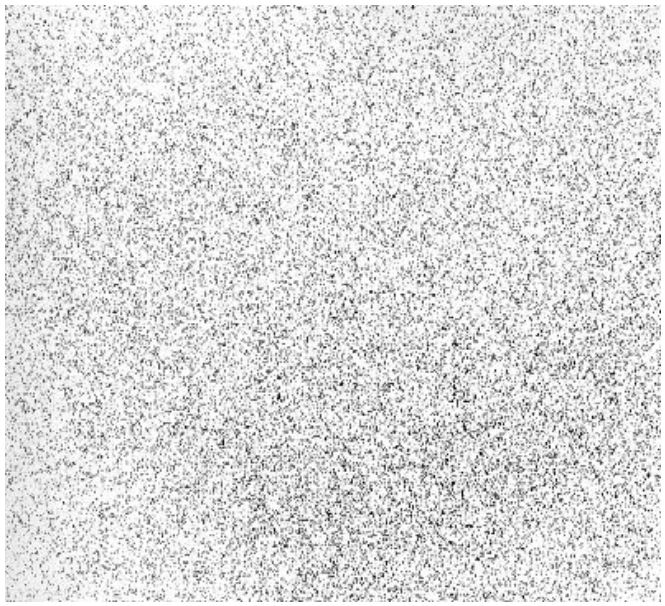
5.

5.

Alice

اولا ترودی خودش میتونه مستقیم با باب در ارتباط باشه چرا این کارو میکنه؟ چون می خواد sequence number که هر بار باب ارسال میکنه رو ببینه ارتباطی بینشون هست یا نه
نکته: sequence number های ما رندوم نیست و sequence number ها یک ارتباط معنایی خاصی با هم دارند ص بعدی نشون میده
پس ترودی sequence number که از باب دریافت میکنه به دست میاره ک احتمالا اگر الان b1 به عنوان sequence number دریافت کردم دفعه بعدی باب چه sequence number ایجاد میکنه و در پیام SYN ACK خودش برمیگردونه ینی از b1 حدس می زنه b2 چی باشه تا اینجا که برسه به b1 می اومد ip ادرس خودشو میذاشت و وقتی رابطه رو دراورد و از b1 می تونست b2 رو بفهمه میاید فیلد ip ادرس رو مال الیس رو قرار میده باب میاد جواب رو واسه الیس می فرسته ولی ترودی طبق کاری که قبلا کرده می تونه حدس بزنه sequence number که باب واسه الیس فرستاده b2 و درست هم حدس زده پس پیام اکی که می فرسته b2+1 است و دیتا رو هم می فرسته و باب اینجا بخاطر این که ACK رو دریافت میکنه و فکر میکنه از سمت الیس و هر داده ای که باب اینجا دریافت بکنه به عنوان داده معتبر از الیس در نظر میگیره
برای این که پیام به الیس نفرسته و نتونه بگه من پیامی ندادم ترودی میاد یک حمله نقض سرویس روی الیس می زنه
پس ترودی هم به باب ازطریق سیستم احراز اصالت ضعیفی که داره حمله میکنه و هم به الیس حمله میکنه و برای الیس اونقدر پیام می فرسته که دیگ پیام باب بهش نرسه
اگر الیس اینجا افلاین باشه دیگه به حمله نقض سرویس نیازی پیاده نمیکنیم و اگر انلاین باشه اون حمله ای که گفتیم می زنیم از نظر محاسباتی و پهنای باند اونقدر الیسو سرگرم بکنه که اون پیام باب منقضی بشه
اشکال پروتکل TCP برای احراز اصالت

TCP Authentication Attack



Random SEQ numbers

Initial SEQ numbers
Mac OS X

- If initial SEQ numbers not very random…
- …possible to guess initial SEQ number…
- …and previous attack will succeed

TCP Authentication Attack

- Trudy cannot see what Bob sends, but she can send packets to Bob, while posing as **Alice**
- Trudy must prevent Alice from receiving Bob's response (or else connection will terminate)
- If **password** (or other authentication) required, this attack fails
- If TCP connection is relied on for authentication, then attack might succeed
- **Bad idea** to rely on TCP for authentication

+ ترودی نمی تواند ببیند باب چه چیزی می فرستد، اما می تواند بسته هایی را برای باب بفرستد، در حالی که خود را به عنوان آلیس نشان می دهد
+ ترودی باید از دریافت پاسخ باب توسط آلیس جلوگیری کند (در غیر این صورت اتصال قطع خواهد شد)
+ اگر رمز عبور (یا دیگر احراز هویت) مورد نیاز باشد، این حمله با شکست مواجه می شود
+ اگر برای احراز هویت به اتصال TCP متکی باشد، ممکن است حمله با موفقیت انجام شود
+ ایده بدی است که برای احراز هویت به TCP اعتماد کنید

نکته: اگر به چنین مکانیزم احراز اصالت یک پسورد ساده اگر بهش ااضافه بکنیم دیگه این حمله کار نمیکنه ینی یک مکانیزیم احراز اصالت مبتنی بر پسورد ضعیف از این مکانیزیم احراز اصالت قوی تر است

Zero Knowledge Proofs

Zero Knowledge Proof (ZKP)

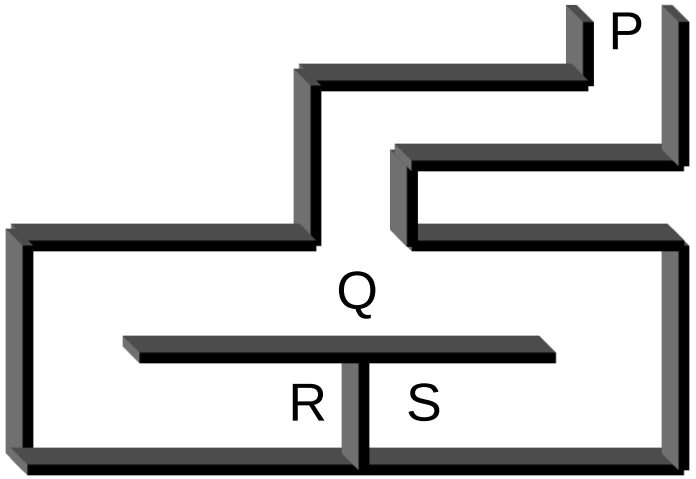
- Alice wants to prove that she knows a secret without revealing **any** info about it
- Bob must verify that Alice knows secret
  - But, Bob gains no information about the secret
- Process is probabilistic
  - Bob can verify that Alice knows the secret to an arbitrarily high probability
- An "interactive proof system"

+ آلیس می خواهد ثابت کند که رازی را می داند بدون اینکه هیچ اطلاعاتی در مورد آن فاش کند
+ باب باید بررسی کند که آلیس راز را می داند: باب باید تایید بکنه که الیس این راز رو میدونه
o اما، باب هیچ اطلاعاتی در مورد راز به دست نمی آورد ینی نباید اطلاعاتی از راز به دست باب برسه
+ فرآیند احتمالی است
o باب می تواند تأیید کند که آلیس راز احتمال زیاد خودسرانه را می داند
+ "سیستم اثبات تعاملی"

اینجا فقط الیس راز رو میدونه و باب چیزی نمیدونه ولی میتونه تایید بکنه (باب)
این پروسه ای که دارند طی می کنند احتمالی است و یک حالت تعاملی داره

Bob's Cave

- Alice knows secret phrase to open path between R and S ("open sarsaparilla")
- Can she convince Bob that she knows the secret without revealing phrase?

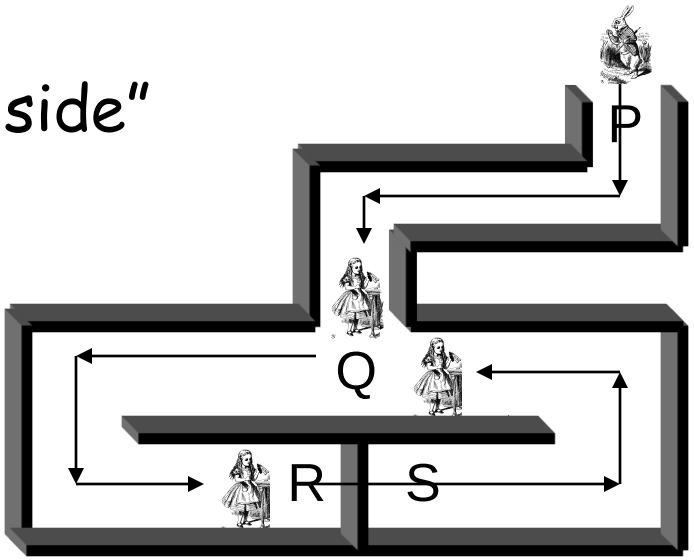


آلیس عبارت مخفی را برای باز کردن مسیر بین R و S می داند ("سارساپاریلا باز")
آیا او می تواند باب را متقاعد کند که راز را بدون فاش کردن عبارت می داند؟

غارباب:
الیس ادعا میکنه اون چیزی که در غارو باز میکنه اینو میدونه و میخواد به باب نشون بده این رازو میدونه بدون اینکه به باب بگه اون رازو
کاری که میکنه اینه که:
الیس روی Q می ایسته و به صورت رندوم انتخاب میکنه بره سمت R یا سمت S
بعد از این که الیس اومد در جای خودش قرار گرفت چه در R و چه در S باب وارد میشه و در نقطه Q قرار میگیره و بعد بلند باب میگه به الیس که از سمت R بیا یا از سمت S اگر الیس سمت R باشه وباب بگه از سمت R که یه راست میاد بالا ینی نیازی نداره درو باز کنه این میشه حالت e=0 در صفحه 66 و در حالتی که e=0 الیس نیازی به دانستن S نداره و اینجا باب برای اینکه بخواد بگه از سمت راست بیا یا سمت چپ میاد سکه می ندازه ینی باز رندومه
ولی اگر الیس در قسمت R باشه و بهش بگه از S بیاد باید الیس درو باز بکنه پس با احتمال 1/2 الیس نیاز داره درو باز بکنه و از سمتی بیاد که باب بهش گفته و با احتمال 1/2 هم نیاز نداره درو باز بکنه
اینجا ترودی هم شانس 1/2 داره پس ایا اساسا روش خوبی است برای اثبات؟
نکت: این پروتکل بارها تکرار میشه که احتمال موفقیت ترودی به شدت کاهش پیدا بکنه
این پروتکل n بار تکرار میشه
اگر اینو n بار اینو تکرار کنیم احتمال این که ترودی بیاد حرف بزنه این احتمال به سمت صفر میل میکنه پس یکبار اجرا برای تایید کردن اصلا کافی نیست
پروتکلی که مبتنی بر این داستان داریم Fiat-shamir است

Bob's Cave

- Bob: "Alice, come out on S side"
- Alice (quietly): "Open sarsaparilla"
- If Alice does not know the secret…



- …then Alice could come out from the correct side with probability 1/2
- If Bob repeats this n times and Alice does not know secret, she can only fool Bob with probability $1/2^n$

Fiat-Shamir Protocol

- Cave-based protocols are inconvenient
  - Can we achieve same effect without the cave?
- Finding square roots modulo N is difficult
  - Equivalent to factoring
- Suppose N = pq, where p and q prime
- Alice has a secret S
- N and $v = S^2 \bmod N$ are **public**, S is **secret**
- Alice must convince Bob that she knows S without revealing any information about S

پروتکل های مبتنی بر غار ناخوشایند هستند
o آیا می توانیم بدون غار به همان اثر برسیم؟
یافتن ریشه های مربع مدول N مشکل است
o معادل فاکتورینگ
فرض کنید N = pq، که p و q اول هستند
آلیس یک اس راز دارد
 و v = S2 mod N عمومی هستند، S مخفی است
آلیس باید باب را متقاعد کند که S را بدون فاش کردن اطلاعاتی درباره S می شناسد

کاری ک میکنه اینه که میاد براساسی که پیدا کردن ریشه دوم در پیمانه N سخت است میاد عمل میکنه و معادل تجزیه است
پس پیمانه خودش رو باتوجه به مسئله تجزیه ایجاد میکنه یکی یک p,q رندوم میگیره و در هم ضرب میکنه
بعد الیس هم یک راز S میخواد بگه میدونه
پس p,q , راز الیس ینی S اینا private هستند و N , s^2 mode N اینا public هستند
الیس میخواد به باب اثبات کنه S رو میدونه بدون این که چیزی از S رو به باب بگه

Fiat-Shamir

Alice
secret S
random r

$x = r^2 \bmod N$

$e \in \{0,1\}$

$y = r \cdot S^e \bmod N$

Bob
random e

- **Public:** Modulus N and $v = S^2 \bmod N$
- Alice selects random r, Bob chooses $e \in \{0,1\}$
- Bob verifies: $\mathbf{y^2 = x \cdot v^e \bmod N}$
  - Note that $y^2 = r^2 \cdot S^{2e} = r^2 \cdot (S^2)^e = x \cdot v^e \bmod N$

کاری که الیس میکنه اینه که یک مقدار رندومی مثل r انتخاب میکنه و به توان دو می رسونه و واسه باب می فرسته و باب اینجا بین صفر و یک ینی همون مرحله سکه انداختن است یک مقداری رو انتخاب میکنه و اینو میده به الیس و الیس اینو حساب میکنه و واسه باب میفرسته بعد y رو به توان دو می رسونه و اگر y با اونی که گفته برابر بود می فهمه الیس راز S رو می دونه

Fiat-Shamir: e = 1

Alice
secret S
random r

$x = r^2 \bmod N$

$e = 1$

$y = r \cdot S \bmod N$

Bob
random e

- **Public:** Modulus N and $v = S^2 \bmod N$
- Alice selects random r, Bob chooses e =1
- If $y^2 = x \cdot v \bmod N$ then Bob accepts it
  - And Alice passes this iteration of the protocol
- Note that Alice must know S in this case

Fiat-Shamir: e = 0

Alice
secret S
random r

$x = r^2 \bmod N$

e = 0

$y = r \bmod N$

Bob
random e

- **Public:** Modulus N and $v = S^2 \bmod N$
- Alice selects random r, Bob chooses e = 0
- Bob must checks whether $y^2 = x \bmod N$
- "Alice" does **not** need to know S in this case!