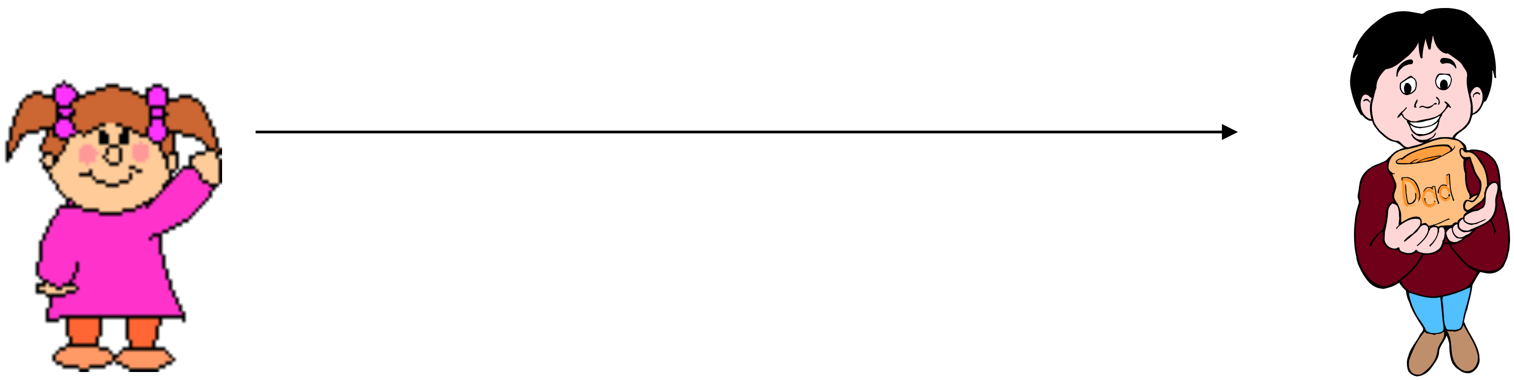
Cristina Nita-Rotaru

CS355: Cryptography

Lecture 17: X509. PGP. Authentication protocols.
Key establishment.

Public Keys and Trust



Public Key: P_A
Secret key: S_A

Public Key: P_B
Secret key: S_B

- **How are public keys stored**
- **How to obtain the public key?**
- **How does Bob know or 'trusts' that P_A is Alice's public key?**

Distribution of Public Keys

- **Public announcement**: users distribute public keys to recipients or broadcast to community at large
- **Publicly available directory**: can obtain greater security by registering keys with a public directory
- Both approaches have problems, and are vulnerable to forgeries

X.509 Authentication Service

- Part of X.500 directory service standards.
- Defines framework for authentication services:
  - Defines that public keys stored as **certificates** in a public directory.
  - Certificates are **issued and signed** by an entity called **certification authority (CA).**
- Used by numerous applications and protocols: SSL, IPSec.
- Started 1988

Public-Key Certificates

- Certificates allow key exchange

  without real-time access to public-key authority
- A certificate binds identity to public key
- Contents signed by a trusted Public-Key or Certificate Authority (CA)
- Can be verified by anyone who knows the public-key authorities public-key
- A commonly used standard to store certificates is PEM.

X.509 Certificates

- Certificates contain:
  - version (1, 2, or 3)
  - serial number (unique within CA) identifying certificate
  - signature algorithm identifier
  - issuer X.500 name (CA)
  - period of validity (from - to dates)
  - subject X.500 name (name of owner)
  - subject public-key info (algorithm, parameters, key)
  - issuer unique identifier (v2+)
  - subject unique identifier (v2+)
  - extension fields (v3)
  - signature (of hash of all fields in certificate)

How to Obtain a Certificate?

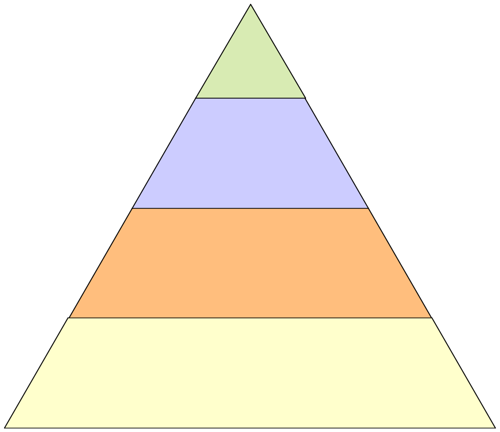
- For a particular application you can define your own CA (libraries like openssl provide the necessary tools)
- Many companies define their own CA.
- Verisign: company that provides certificates; commercial companies obtain certificates;
- Private key remains secret and certificate must be accessible.
- Example: see certificates accepted by your browser, if you use netscape: preferences/ security and privacy/certificates

Validity of Certificates

- Certificates are valid if:
  - Signature of CA verifies
  - Dates of the certificate are valid
  - Certificate was not revoked
- Certificates can be revoked before expiration if
  - user's private key is compromised
  - user is no longer certified by this CA
  - CA's certificate is compromised
- CA maintains a list of revoked certificates: Certificate Revocation List (CRL)
- Users should check certificates with CA's CRL

CA Hierarchy

- If everybody has the same CA then they are assumed to know its public key, so they can verify each other's certificate. Not scalable.
- Other approach: entities have different CAs; in this case CAs how is a certificate verified?
  - CAs must form a hierarchy
  - certificates linking members of hierarchy are used to validate other CAs
  - each CA has certificates for clients (forward) and parent (backward)
  - each client trusts parents certificates



CAs and Trust

- Certificates are trusted if signature of CA verifies
- Chain of CA's can be formed, head CA is called root CA
- In order to verify the signature, in the end the public key of the root CA should be obtain. When is that valid?
- "You just trust the root CA".
- TRUST is CENTRALIZED (one CA) or HIERARCHICAL (more CAs.)

Problems with X509

- Management of certificates
- Assumptions about validity of certificates:
  - detection of secret key disclosure
  - time delay for certificate revocation
  - time delay for distribution of revoked certificates
  - amount of data distributed periodically by CA

Problems with X509 (2)

- CRLs have several problems
  - Protocols must check CRLs to make sure that the certificate is still valid
  - In practice protocols do not really check CRLs, delay between revocation and detection of revocation
  - CRL is not suitable for time-critical applications
  - time-validity of CRL is typically 24 hours
  - Validity of certificates is usually years

Detection of Secret Key Disclosure

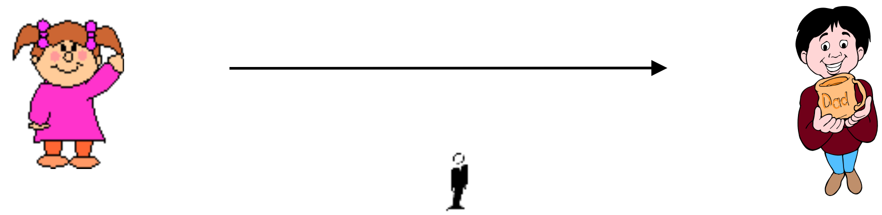
- Time between disclosure and detection may be in hours or days, time needed for abuse may be counted in milliseconds
- Owner is responsible for private key usage until requesting CA to revoke appropriate certificate
- There is no trusted way to identify place or time of signature creation

PGP

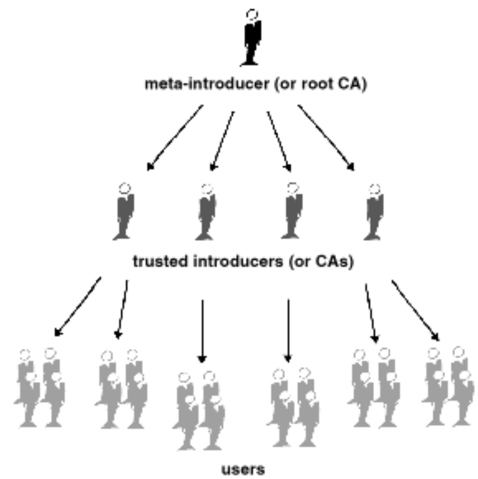
- PGP (Pretty Good Privacy) is a secure email application
- Mail is encrypted and signed using public keys
- What's different? The way the keys are authenticated, trust about the keys is built.
- Trust is not centralized.
- **http://www.pgpi.org/**

Trust Models

- Direct Trust



- Hierarchical trust



- Web of trust: combination of both

Cristina Nita-Rotaru

PGP Web of Trust

- Any user can act as a CA
- Certificate is only valid if the receiving party recognize the validator as a trusted introducer
- Each user stores:
  - Its own public/private keys
  - Keys of entities that interacts with
  - whether or not the user considers a particular key to be valid
  - the level of trust the user places on the key that the key's owner can serve as certifier of others' keys

Problems

- Key revocation of a key, a user needs to issue a revoked certificate and then distribute it as broad as possible.
- Does not scale for large, open communities
- Does not really accomodate for more formalised security needs, for instance for non-repudiation purposes towards a third party

Authentication

- **Entity authentication (identification)**: the process whereby one party is assured of the identity of a second party involved in a protocol an that the second has actually participated.
- **Data source authentication**: represents an indication about the source of the data.

Requirements of Identification Protocols

- Requirements of identification protocols
  - for honest prover A and verifier B, A is able to convince B
  - no other party can convince B
  - in particular, B cannot convince C that it is A
- Kinds of attackers
  - passive and replay
  - active, man in the middle
  - the verifier

Properties of Identification Protocols

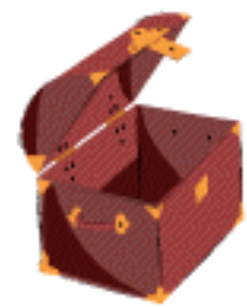
- Reciprocity of identification (one -way or mutual)
- Computational efficiency (encryption, signing)
- Communication efficiency (communication rounds, messages)
- Involvement of a third party
- Nature of trust in the third party
- Storage of secrets

Authentication Using Fixed Passwords

- Client authenticates to a server using a password.
- Passwords must be kept in encrypted password files or as digests
- Strengthen passwords by “salting”
- Passphrases, more complex passwords
- Attacks:
  - Replay of fixed passwords
  - Exhaustive password search
  - Password-guessing and dictionary attacks

Unix crypt Algorithm

- Used to store Unix passwords
- Information stored is /etc/passwd is:
  - Iterated DES encryption of 0 (64 bits), using the first 8 characters of the password as key
  - 12 bit random salt taken from the system clock time at the password creation
- Why use the salt: to alter the expansion function E of DES, to defend against attacks on DES using off-the-shelf hardware that can crack DES



Lamport's One-Time Password

Stronger authentication that password-based

- One-time setup:
  - A selects a value w, a hash function H(), and an integer t, computes $w_0 = H^t(w)$ and sends w_0 to B
  - B stores w_0
- Protocol: to identify to B for the i^{th} time, $1 \le i \le t$
  - A sends to B: A, i, $w_i = H^{t-i}(w)$
  - B checks $i = i_A$, $H(w_i) = w_{i-1}$
  - if both holds, $i_A = i_A+1$

Challenge-Response Protocols

- Goal: one entity authenticates to other entity proving the knowledge of a secret, 'challenge'
- Time-variant parameters used to prevent replay, interleaving attacks, provide uniqueness and timeliness : nounce (used only once)
- Three types:
  - Random numbers
  - Sequences
  - Timestamp

Challenge-Response Protocols

- **Random numbers**:
  - pseudo-random numbers that are unpredictable to an adversary;
  - vulnerable to birthday attacks, use larger sample;
  - must maintain state;
  - do not prevent interleaving attacks (parallel sessions)
- **Sequences**:
  - serial number or counters;
  - long-term state information must be maintained by both parties+ synchronization
- **Timestamp**:
  - provides timeliness and detects forced delays;
  - requires synchronized clocks

Challenge-Response Protocols Using Digital Signatures

- unilateral authentication with timestamp

  $A \rightarrow B$: $cert_A, t_A, B, S_A(t_A, B)$
- unilateral authentication with random numbers

  $A \leftarrow B$: r_B

  $A \rightarrow B$: $cert_A, r_A, B, S_A(r_A, r_B, B)$
- mutual authentication with random numbers

  $A \leftarrow B$: r_B

  $A \rightarrow B$: $cert_A, r_A, B, S_A(r_A, r_B, B)$

  $A \leftarrow B$: $cert_B, A, S_B(r_B, r_A, A)$

Attacks: Examples

- E1: "Man-in-the-middle" attack on unauthenticated DH
- E2: Reflection attack

Protocol: A and B authenticate to each other

(1) $A \rightarrow B : r_A$

(2) $B \rightarrow A : E_k(r_A, r_B)$

(3) $A \rightarrow B : r_B$

Attack: E wants to trick A to accept him as B

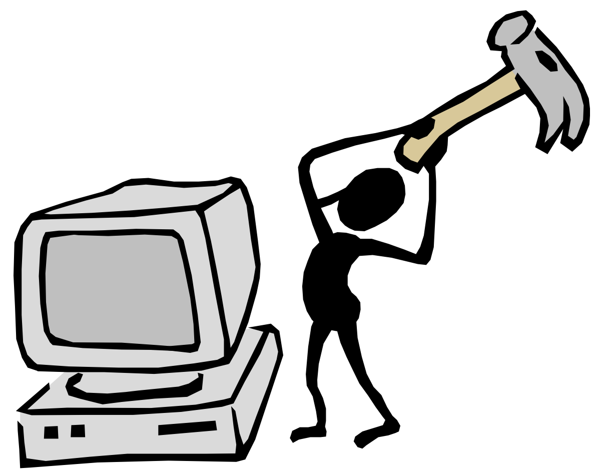
(1) $A \rightarrow E : r_A$

(2) $E \rightarrow A : r_A$: Starting a new session

(3) $A \rightarrow E : E_k(r_A, r_A')$: Reply of (2)

(4) $E \rightarrow A : E_k(r_A, r_A')$: Reply of (1)

(5) $A \rightarrow E : r_A'$; this concludes session started with (1)



AUTHENTICATION RELIES ON THE SECRECY OF KEY K

Attacks: Examples (cont.)

- E3: Interleaving attacks (parallel sessions)

  Protocol

  (1) $A \rightarrow B : r_A$

  (2) $B \rightarrow A : r_B, S_B(r_B, r_A, A)$

  (3) $A \rightarrow B : r_A', S_A(r_A', r_B, B)$

  Attack: E wants to pass as A to B

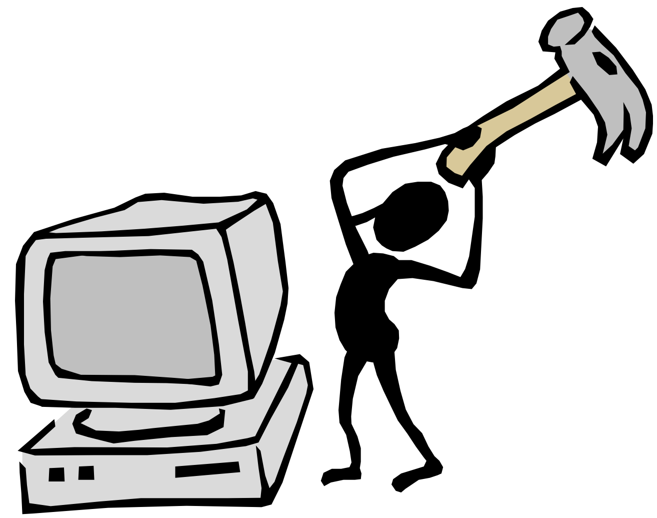
  (1) $E \rightarrow B : r_A$

  (2) $B \rightarrow E : r_B, S_B(r_B, r_A, A)$

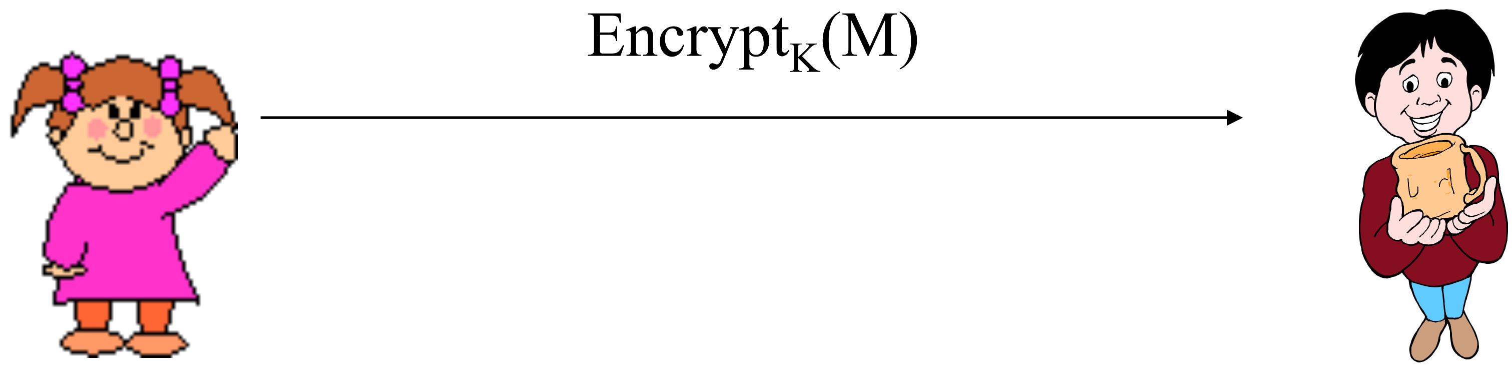
  (3) $E \rightarrow A : r_B$

  (4) $A \rightarrow E : r_A', S_A(r_A', r_B, B)$

  (5) $E \rightarrow B : r_A', S_A(r_A', r_B, B)$



Need for Key Establishment



$C = Encrypt_K(M)$ $M = Decrypt_K(C)$

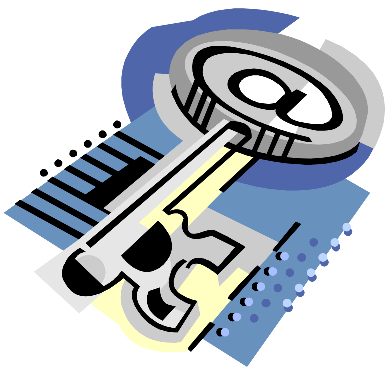
- Alice and Bob share a secret key K
- **How to establish the shared key?**
- **How to refresh it (not a good idea to encrypt a lot of data with the same key)**

Long-Term Key vs. Session Key

- **Session key**: temporary key, used for a short time period.
- **Long-term key**: used for a long term period, sometimes public and secret key pairs used to sign messages.
- Using session keys to:
  - limit available cipher-text encrypted with the same key
  - limit exposure in the event of key compromise
  - avoid long-term storage of a large number of distinct secret keys
  - create independence across communications sessions or applications

Key Establishment

- **Key pre-distribution**: keys are distributed off-line
- **Dynamic shared key establishment**: protocols that define on-line key establishment
- **Key establishment**: process to establish a shared secret key available to two or more parties;
  - key transport: one party creates, and securely transfers it to the other(s).
  - key agreement: key establishment technique in which a shared secret is derived by two (or more) parties



Issues in Key Establishment

- Need and type of the authentication: unilateral vs. mutual
- Key control: key distribution vs. key agreement
- Efficiency: communication (number of message and communication rounds) and computation (exponentiations and digital signatures) costs
- Two ways to achieve:
  - using symmetric encryption
  - using public key encryption
- Use of trusted third party (TTP):
  - on-line/off-line/no third party
  - degree of trust required in a third party

Basic Key Transport Protocol

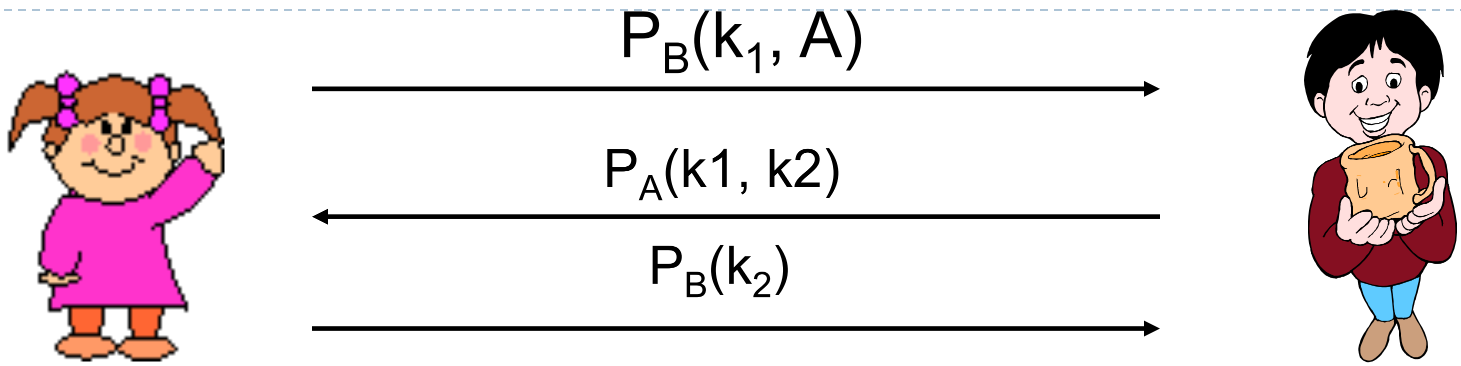
- Assumes a long term symmetric key K shared between A and B
- Basic: new key is k_A

$$A \rightarrow B: E_K(k_A)$$

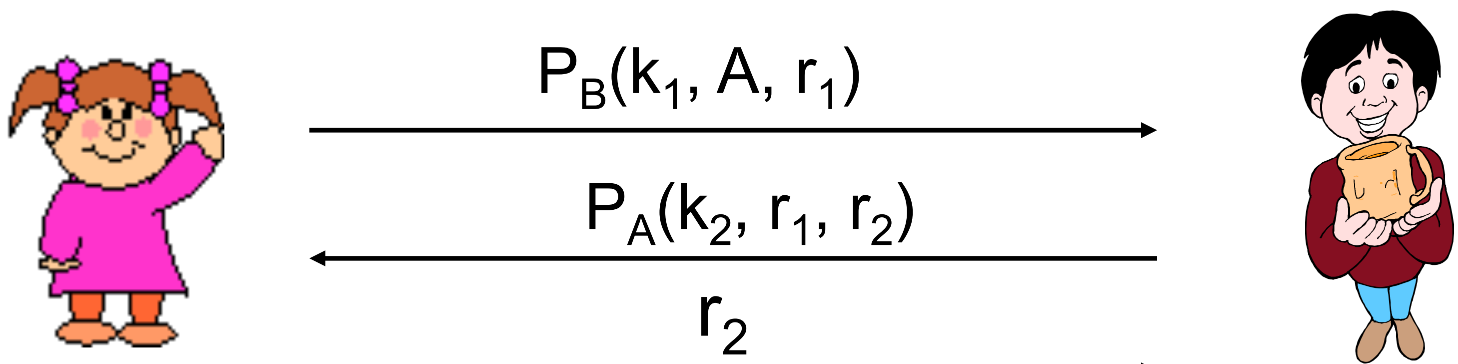
- Prevents replay: new key is r_A

$$A \rightarrow B: E_K(k_A, t_A, B)$$

Needham-Schroeder Public Key Protocol



- P_A and P_B denote public keys;
- A and B distribute keys k_1 and k_2



Key Transport: Combining Public Key Encryption and Digital Signature

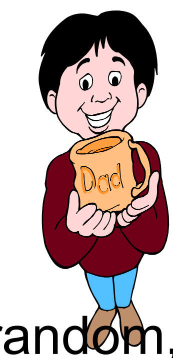
- Encrypting signed keys:
  - $A \rightarrow B$: $P_B(k, t_A, S_A(B, k, t_A))$
  - Problem: Data for encryption is too large
- Encrypting and signing separately
  - $A \rightarrow B$: $P_B(k, t_A), S_A(B, k, t_A)$
  - Acceptable only if no information regarding plaintext data can be deduced from the signature
- Signing encrypted keys
  - $A \rightarrow B$: $t_A, P_B(A, k), S_A(B, t_A, P_B(A, k))$
  - Can provide mutual authentication with two messages(timestamps) or three messages(challenge-response)

Key Agreement: Diffie-Hellman Protocol

- Key agreement protocol, both A and B contribute to the key
- Setup Z_n, n prime and g generator, n and g public.

g^a mod n $\rightarrow$

$\leftarrow$ g^b mod n



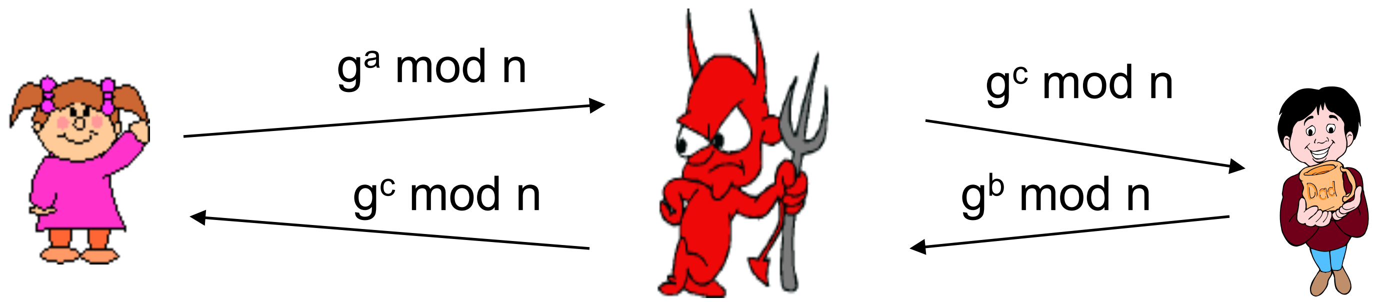
Pick random, secret a
Compute and send g^a mod n

$K = (g^b \bmod n)^a = g^{ab} \bmod n$

Pick random, secret b
Compute and send g^b mod n

$K = (g^a \bmod n)^b = g^{ab} \bmod n$

Station-to-Station (STS)



Alice computes g^{ac} mod n and Bob computes g^{bc} mod n !!!

- Provides mutual entity authentication