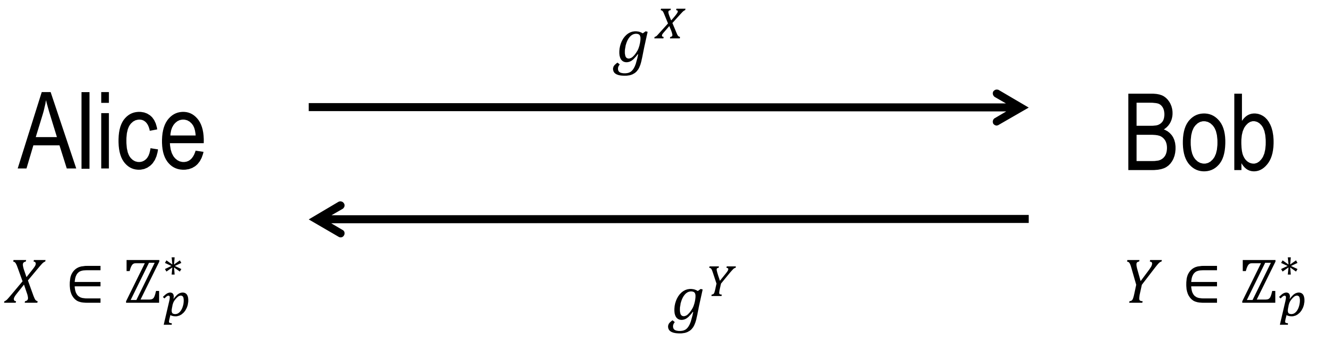
Key Negotiation, SSL, PKI

Cryptography and Protocols

Andrei Bulatov

Man-in-the-Middle Attack



g^X

Alice

Bob

$X \in \mathbb{Z}_p^*$

g^Y

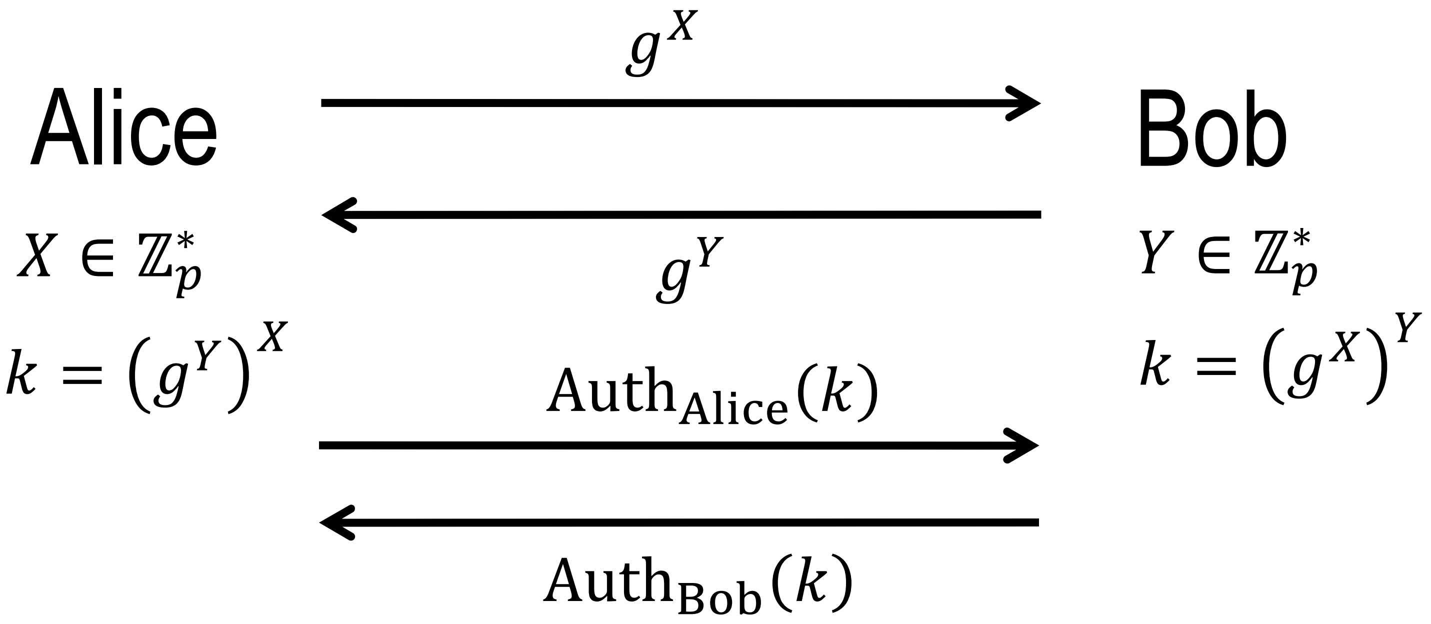
$Y \in \mathbb{Z}_p^*$

$k = \left(g^Y\right)^X$

$k = \left(g^X\right)^Y$

- Man-in-the-Middle attack
- There is no way to fix this protocol unless Alice and Bob know something about each other

A Better Protocol



Alice

$X \in \mathbb{Z}_p^*$

$k = \left(g^Y\right)^X$

g^X

g^Y

$\mathrm{Auth}_{\mathrm{Alice}}(k)$

$\mathrm{Auth}_{\mathrm{Bob}}(k)$

Bob

$Y \in \mathbb{Z}_p^*$

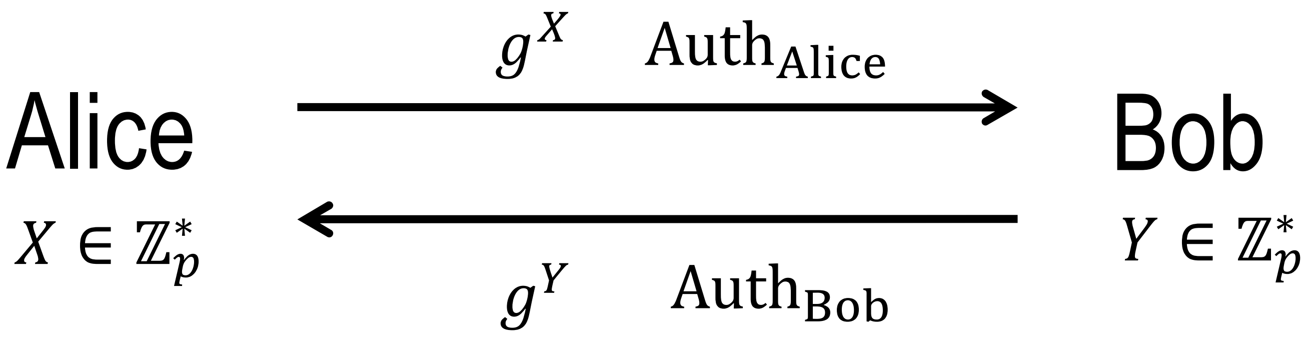
$k = \left(g^X\right)^Y$

- Authentication can be done by a MAC using a secret key or by a digital signature

A Better Protocol: Problems

- There are problems:
- It uses 4 messages, while 3 are enough
- Session key is used for authentication
- The authentication messages are too similar. When using MAC, Bob can resend Alice’s message
- Prime p is a constant

Second Attempt



Alice

$X \in \mathbb{Z}_p^*$

$g^X \quad \text{Auth}_{\text{Alice}}$

$g^Y \quad \text{Auth}_{\text{Bob}}$

Bob

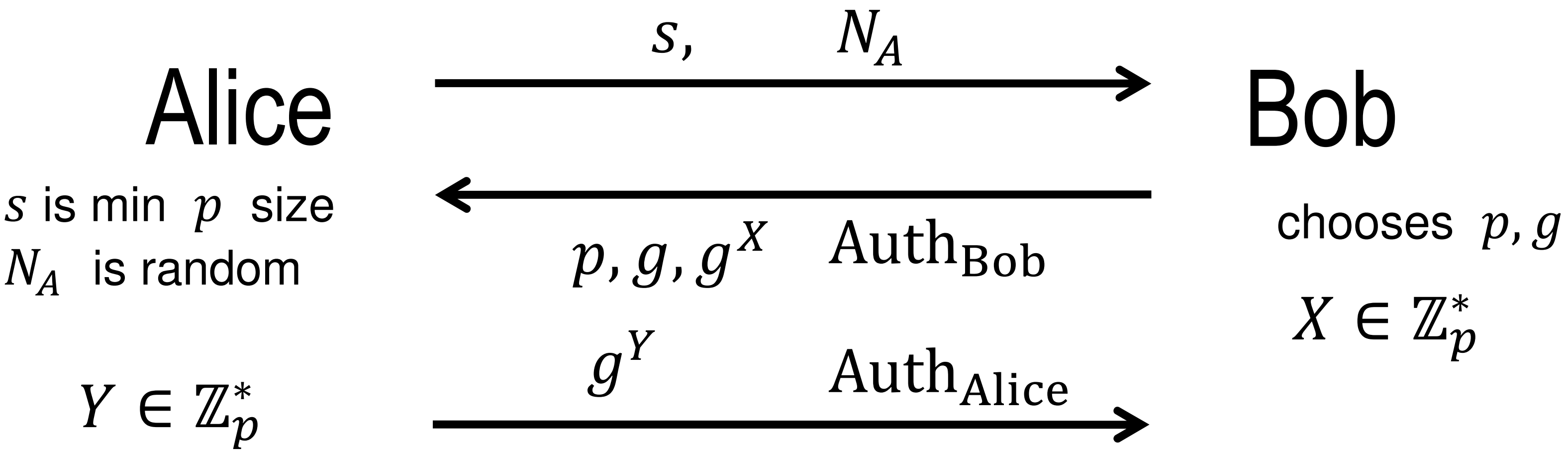
$Y \in \mathbb{Z}_p^*$

- Here $\text{Auth}_{\text{Alice}}$, Auth_{Bob} mean authentication of all messages previously sent or received. Thus here we have
  $\text{Auth}_{\text{Alice}}(g^X)$ $\text{Auth}_{\text{Bob}}(g^X, g^Y)$

Second Attempt: Problems

- What if Bob is not satisfied with the prime p used in the protocol?
- Bob is not sure he is talking to Alice, replay attack
- p is still a constant

Final Version



Alice

Bob

$s,\quad N_A$

s is min p size

N_A is random

chooses p, g

$X \in \mathbb{Z}_p^*$

$p, g, g^X \quad \mathrm{Auth}_{\mathrm{Bob}}$

$g^Y \quad \mathrm{Auth}_{\mathrm{Alice}}$

$Y \in \mathbb{Z}_p^*$

$$k = SHA\text{-}256\left(\left(g^Y\right)^X\right) \qquad k = SHA\text{-}256\left(\left(g^X\right)^Y\right)$$

- Alice and Bob check authentications

SSL / TLS

- Consists of 3 parts:
  - negotiation for algorithm support
  - key exchange and authentication
  - symmetric cipher encryption and message authentication
- The first two parts are called the handshaking protocol
- We mostly consider the second part

SSL / TLS

- ClientHello: the available protocol version, a random number, a list of suggested primitives
- ServerHello: the chosen protocol version and primitives, a random number, a session id
- Server’s Certificate message (RSA public key or a digital signature)
- ServerHelloDone: indicating it is done with handshake negotiation.
- ClientKeyExchange: PreMasterSecret, public key, or certificate
- Client and Server use the random numbers and PreMasterSecret to compute a common secret, called the "master secret".
- Client sends a ChangeCipherSpec record, indicating that encrypting starts

SSL / TLS (cntd)

- Client sends an encrypted Finished message, containing a hash and MAC over the previous handshake messages.
- Server will attempt to decrypt the Client's Finished message, and verify the hash and MAC. If the decryption or verification fails, the handshake is considered to have failed and the connection should be torn down.
- Server sends a ChangeCipherSpec and its encrypted Finished message, and the Client performs the same decryption and verification.
- At this point, the "handshake" is complete and the Application protocol is enabled

Certificates: PKI

- PKI stands for Public Key Infrastructure.
- The idea is to prevent Man-in-the-Middle attacks by certifying public keys of the parties
- Note that the Man-in-the-Middle attack cannot be prevented unless the parties have some information about each other
- Such information is usually a certificate issued by a CA, Certifying Agency, such as VeriSign
- A certificate is usually a message like ``Public key PK belongs to Alice'' signed by the digital signature of the CA
- The system of CAs and certificates is called PKI

Certificates: PKI (cntd)

- How can we verify the digital signature of CA?
- There are 2 types of CA, and PKIs
- The first one is local: a company for employees, a bank for clients, etc.
- The second type is universal
- For local CAs the problem of verification is easy
- Digital signatures of universal CAs are available from software manufactures. So if a certificate is verified by an implementation inside Windows, it can ask Microsoft for a CA’s signature

Problems with PKI

- Names, clients identification (not so serious for local CAs)
- Authority / Trust (not so serious for local CAs)
- Revocation. Sometimes certificates should be revoked

Overview of Cryptography

	Symmetric crypto	Asymmetric crypto
Secure system	Pseudorandom generator + stream cipher	----------------------
CPA-secure system	Pseudorandom function + stream cipher Pseudorandom permutation + block cipher Theory primitives: BBS, tree-like comput. Practice: ad-hoc, DES, AES	Trapdoor permutations + tons of precautions RSA, Rabin, ElGamal, Elliptic curves Theory=Practice: Number theory and such
Message auth. CMA-security	MACs Theory primitives: pseudorandom function Practice: hash functions, HMAC	Digital signature Theory primitives: signature chains, commitments, OWP Practice: inverse trapdoor permut. + hash functions
CCA-secure system	CPA + Message authentication	CPA + Digital signature Random oracle model
Protocol	Secure channel Secret key assumed	Handshaking stage Public key/digital signature assumed PKI