

Strong Password-based Downloading of Credentials

Radia Perlman

Sun Microsystems

radia.perlman@sun.com

Goal

- “The Network Is the Computer”
- WSs have appropriate software, but no user-specific config info
- “Log in” from any WS, get everything necessary to reconstruct my environment
 - private key, trust anchors, browser bookmarks and cookies, fonts, etc.

Why Special Protocol Necessary for Private Key

- Want private key (encrypted with password) stored in the directory (or some server)
- Want to send it to user's WS
- Can't be world-readable--subject to off-line dictionary attack
- Can't have traditional ACL protection...I can't prove I'm Radia unless I know my private key!

Rest of context

- Once I know my private key
 - information that needs to be integrity protected, like trust anchors, can be retrieved from directory, signed with my private key
 - information that needs to be kept secret (perhaps browser bookmarks) can be encrypted with my public key

Current solution

- Why not just use SSL?
 - Server sends client its cert
 - Client verifies cert using configured trust anchors
 - Establishes secure tunnel
 - Client sends name/password
- Problem: if have bad trust anchors configured, you've given away your pwd!

Why Passwords?

- *“Humans are incapable of securely storing high-quality cryptographic keys, and they have unacceptable speed and accuracy when performing cryptographic operations. They are also large, expensive to maintain, difficult to manage, and they pollute the environment. It is astonishing that these devices continue to be manufactured and deployed, but they are sufficiently pervasive that we must design our protocols around their limitations.”*

– Network Security: Private Communication in a Public World, by Kaufman, Perlman, Speciner

Copyright 2000 Sun Microsystems, Inc.

Off-line Dictionary Attack

Alice

Bob

compute W from pwd

store $W=h(\text{pwd})$

I'm Alice



Challenge= R



$f(W,R)$



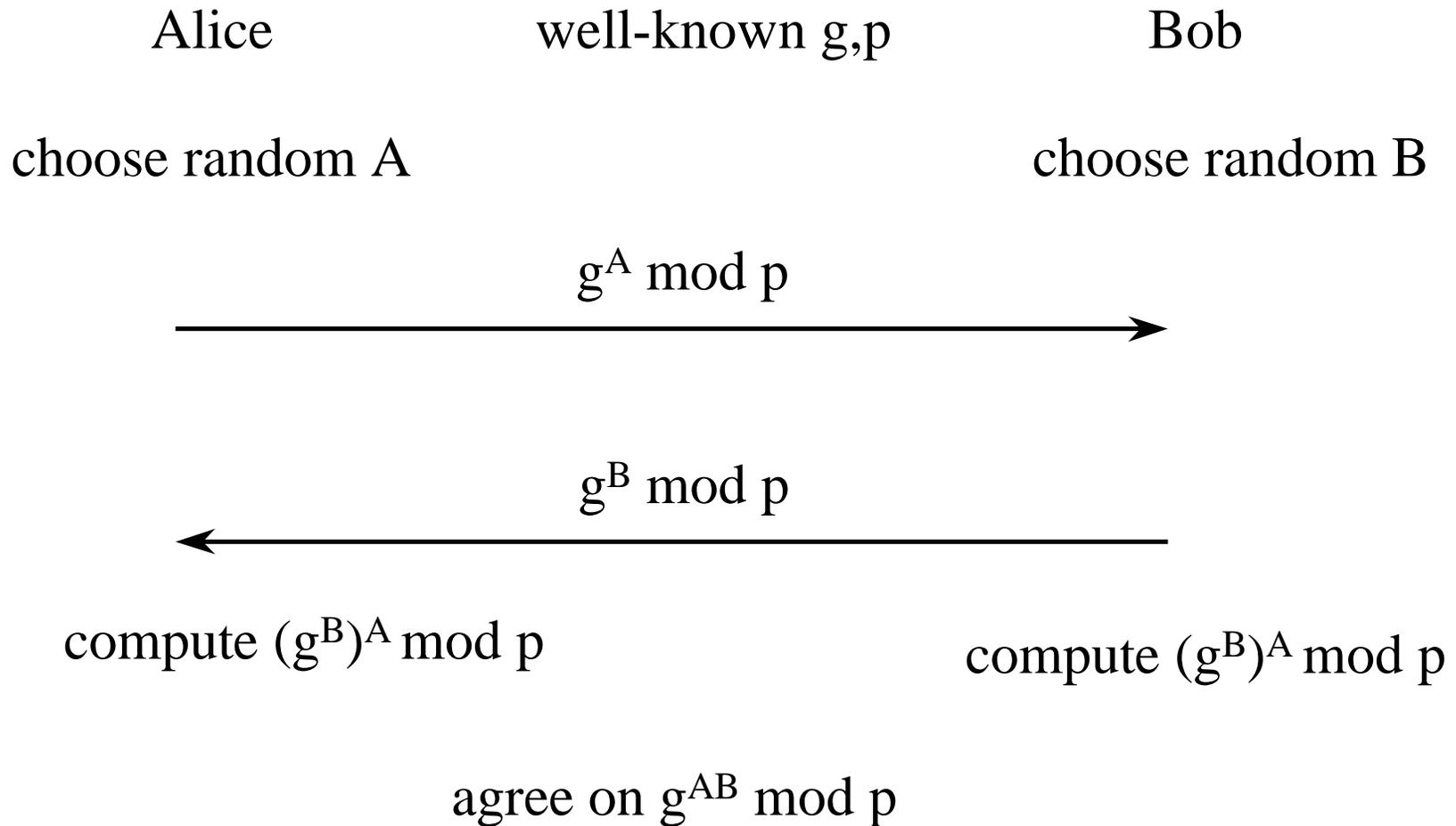
An Intuition for Diffie-Hellman

- Allows two individuals to agree on a secret key, even though they can only communicate in public
- Alice chooses a private number and from that calculates a public number
- Bob does the same
- Each can use the other's public number and their own private number to compute the same secret
- An eavesdropper can't reproduce it

Why is D-H Secure?

- We assume the following is hard:
- Given g , p , and $g^X \bmod p$, what is X ?

Diffie-Hellman



Diffie-Hellman and Strong Password Authentication

- The transmitted D-H numbers look just like random numbers
- Various algorithms for leveraging D-H for strong password authentication
 - EKE (Bellovin-Merritt): encrypt D-H number with $W=h(\text{pwd})$
 - SPEKE (Jablon): replace g with W
 - others...

EKE

Alice

Bob

compute W from pwd

store W

$\{g^A \bmod p\}$ encrypted with W



$g^B \bmod p$



do mutual authentication using g^{AB}



SPEKE

Alice

Bob

compute W from pwd

store W

$W^A \bmod p$



$W^B \bmod p$

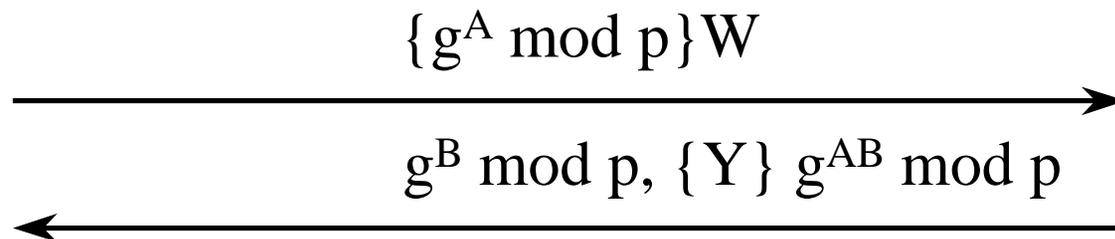


do mutual authentication using W^{AB}

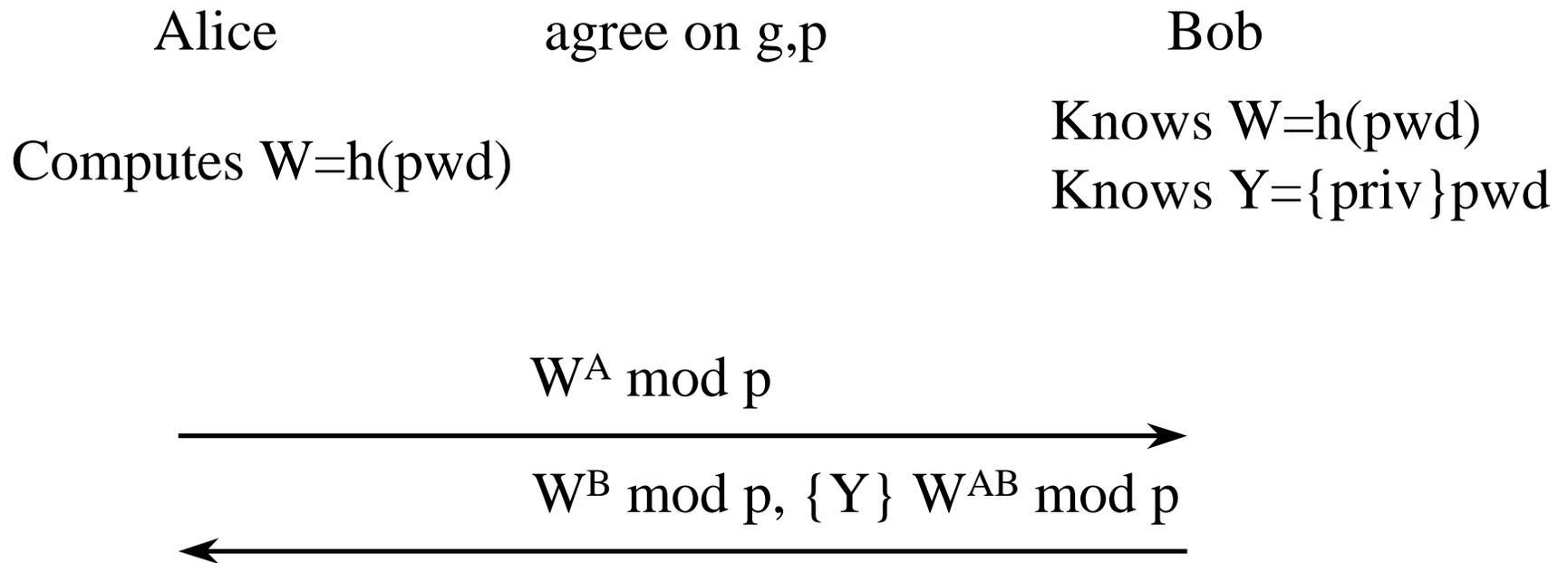


(Perlman/Kaufman) private key download protocol (EKE-based)

Alice	agree on g, p	Bob
Computes $W = h(\text{pwd})$		Knows $W = h(\text{pwd})$ Knows $Y = \{\text{priv}\} \text{pwd}$



SPEKE-based



Perlman/Kaufman Protocol

- Similar in spirit to EKE and SPEKE
 - (to our actual knowledge) unencumbered
 - Internet draft presented at last IETF
 - draft-perlman-strong-pass-00.txt
- Slow at client
- As fast or faster at server. Server's performance more important than client's
- Idea: generate p from W

Basic Protocol

Alice

Bob

compute p from name,pwd

store p, Y, pub

$2^A \bmod p$



$2^B \bmod p, \{ Y \} 2^{AB} \bmod p$



How to Compute p before user expires

- We'd like a safe prime p
 - $(p-1)/2$ also prime
- Want 2 to be a generator (pick $p \equiv 3 \pmod{8}$)
- With secret moduli, can use smaller p
 - have to break Diffie-Hellman *per pwd guess*
 - 8000-MIP-year estimate with 500-bit p 's
 - 10 seconds (400 Mhz CPU) to find 500-bit safe p
- Use Schiller "hint" (optional extra char(s) to cut down search time for p)

Leaking Information

- Have to implement these schemes carefully or else information leaked
- For instance, if implement EKE naively, if decrypt $\{g^A \bmod p\}W$, with guessed $W=h\{\text{guessed pwd}\}$ and the answer is $> p$, can rule out that pwd
- If p is not close to a power of 2, eliminate almost half the passwords each time

Avoid Leaking Information in our scheme

- Timing attacks (therefore, need to generate p before contacting server)
- “small exponent” attack: can do dictionary attack after sending $2^x < p$ (refuse 100...00)
- throw away A if 2^A bigger than smallest possible p . Make this unlikely by choosing p 's with top 65 bits=1111...1110, discard A if top 64 bits of $2^A =$ all 1's

Conclusions

- Other than private key, other security context can be stored in directory, encrypted and/or integrity protected with user's key
- Private key requires special protocol, and can be built on many different “strong password” schemes
- An unencumbered version exists