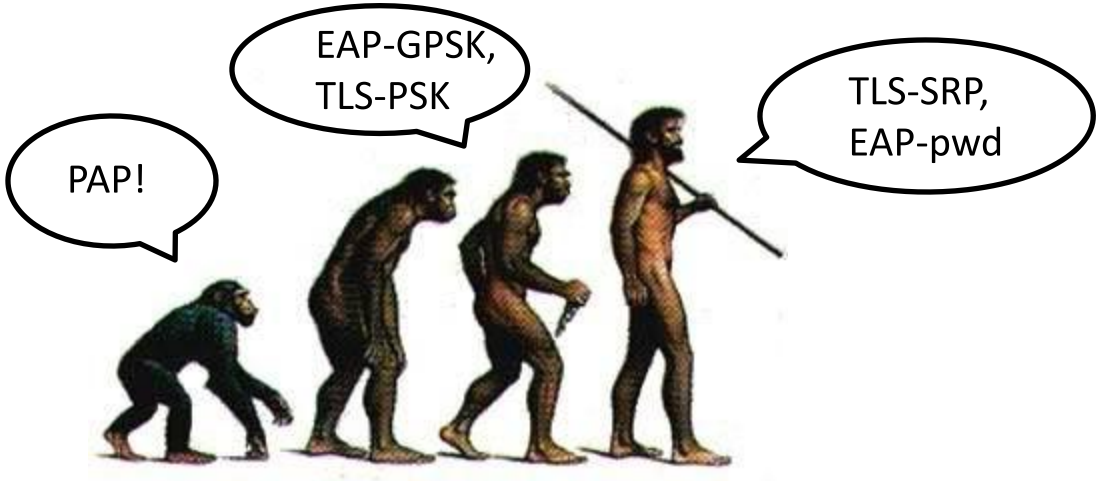
# Dragonfly: A PAKE Scheme

Dan Harkins

IETF 83

Paris, France

# The Rise of Password Protocols in the IETF




- Plaintext passwords (1986 to 1995 or so)
  - PAP-like exchange– completely broken
  - Outlawed by Jeff Schiller
- Password derived data (90s to present)
  - Transmit a hash of the password with nonces– susceptible to dictionary attack
  - Still used today (EAP-GPSK, TLS-PSK, IKE PSK, etc)
- PAKE scheme (2007 - ???)
  - Use a zero-knowledge password protocol– secure!
- *Protocols that are susceptible to dictionary attack are on the Standards Track while those that are resistant to dictionary attack are Informational!*

# Uses for PAKEs

- Certificate-less HTTPS
  - Mitigates the popular and insecure self-signed cert + PAP
  - No more captive portal
  - No need to rely on 3rd party to ensure secure connection
- Robust, misuse-resistant, security
  - Eliminates the need for requiring long, random binary shared secrets <wink, wink> with PSK-based schemes
  - Realistic security in most probable deployment
- Parlay a simple token into a user/device cert
- Any commodity device with a user-interface for configuration that must communicate over a network
  - Most people don’t understand certificates; expecting people to provision their devices with a certificate is naïve
  - Ma and Pa Kettle do not have security clue

# What does this have to do with CFRG?

- There is resistance to PAKEs in the IETF
  - Questions about security always come up
  - Resistance results in promulgation of protocols that are insecure in their most likely usage
- CFRG can help vet PAKEs to allow WGs to have more confidence in adopting them
  - For example, ….

# A Key Exchange Called “dragonfly”

- Yet another PAKE? Yes
- Motivation
  - Symmetric, true peer-to-peer protocol (either side can initiate and both can initiate simultaneously)
  - Use both ECC and FFC and not require special domain parameter sets
  - Don’t bind a user to one particular domain parameter set
  - No IPR issues
- None of the existing schemes were appropriate
- It’s a fun problem to work on too

- Commit then confirm protocol
  - A party may *commit* at any time
  - A party *confirms* after both it *commits* and its peer *commits*
  - A party *accepts* authentication after a peer *confirms*
  - The protocol successfully *terminates* after both parties *confirm*

<u>Assuming</u>:

- **H**() is a secure PRF
- **f**($v$) is a deterministic mapping of string $v$ to an element in G

<u>Given</u>:

- group G = {generator g, prime p, order q [, a, b]}
- a password chosen at random from a pool

Alice and Bob first generate a password-derived element in G:

$$PE = \mathbf{f}(\mathit{password})$$

- Commit phase
  - Exchange scalars and elements
  - Generate shared secret

**Alice**

rnd-a, msk-a <--- random()

scalar-a = (rnd-a + msk-a) mod q

element-a = $PE^{-\text{msk-a}}$

**Bob**

rnd-b, msk-b <--- random()

scalar-b = (rnd-b + msk-b) mod q

element-b = $PE^{-\text{msk-b}}$

$$(PE^{\text{scalar-b}} * \text{element-b})^{\text{rnd-a}} \bmod p = ss = (PE^{\text{scalar-a}} * \text{element-a})^{\text{rnd-b}} \bmod p$$

- Confirm phase
  - Generate master key, key confirmation key
  - Exchange confirm messages

Alice — Bob

KCK | MK = KDF(ss, “some cruft”, (scalar-a + scalar-b) mod q)

confirm-a = **H**(KCK, scalar-a | scalar-b | element-a | element-b)

confirm-b = **H**(KCK, scalar-b | scalar-a | element-b | element-a)

If confirms are verified, exchange succeeds (use MK), else it fails

- Specified in many protocols
  - IEEE 802.11-2012 for authentication between wireless devices (client and AP, or nodes in mesh and ad hoc networks), SAE
  - EAP, RFC 5931
  - IKE, draft-harkins-ipsecme-spsk-auth
  - TLS, draft-harkins-tls-pwd

- Is this scheme secure?
  - Is the probability that an adversary can break the protocol less than the probability of the adversary guessing the password outright?
  - Does the adversarial advantage grow through *interaction* and not through *computation*?
  - Does any information (except the knowledge that a single guess is correct or incorrect) leak as a result of running the protocol?

# Secure Against Passive Attack

- CDH problem:
  - given $(g^a, g^b, g)$
  - produce $g^{ab}$
- dragonfly algorithm:
  - given $(ra+ma, PWE^{-ma}, rb+mb, PWE^{-mb}, PWE)$
  - produce $PWE^{ra*rb}$
- Reduction:
  - generate random r1, r2
  - Give attacker $(r1, g^a, r2, g^b, g)$ to produce $g^{(r1+a)*(r2+b)}$
  - But $g^{(r1+a)*(r2+b)} / ((g^a)^{r2} * (g^b)^{r1} * g^{r1*r2}) = g^{ab}$ !
- Conclusion:
  - Successful attack against dragonfly would solve CDH problem, which is computationally infeasible

# Secure Against Active Attack?

- “doesn't seem likely that the protocol can be proven secure”– Jonathan Katz
- Random oracle model
  - assume no key confirmation step in dragonfly, just scalar and element exchange
  - adversary performs MitM, adding 1 to one side’s scalar
  - adversary issues “reveal” query to obtain secrets of both sides
  - off-line dictionary attack is now possible
- This is too contrived to worry about as a real attack against dragonfly but it is a problem with a formal proof of security (at least in Random Oracle model)
- Can this protocol be proven secure? Help.