New Methods for Password Authenticated Key Exchange

Craig Gentry

cgentry@cs.stanford.edu

Stanford University

Joint work with Phil MacKenzie (Google), Zulfikar Ramzan (Symantec) while we were all at DoCoMo USA Labs

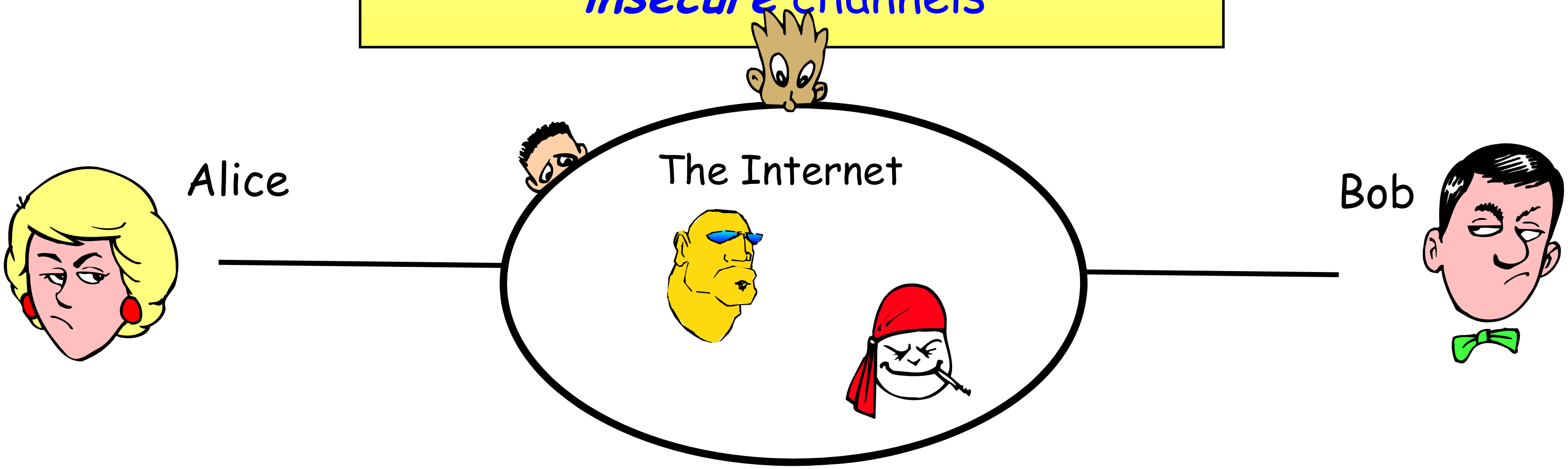
Stanford Security Workshop 2006

Outline

- What is password authenticated key exchange (PAKE)?
- Prior work, including patents
- Our results
  - ✓ A new PAKE scheme (called "OPAKE")
  - ✓ Theoretically: it basically uses "oblivious-transfer" as a building block.
  - ✓ Practically: it has competitive performance, and seems to "design around" pesky patents
- Conclusions

The Basics: Major Goal of Crypto

One Major Goal of Cryptography:
secure communications over
insecure channels

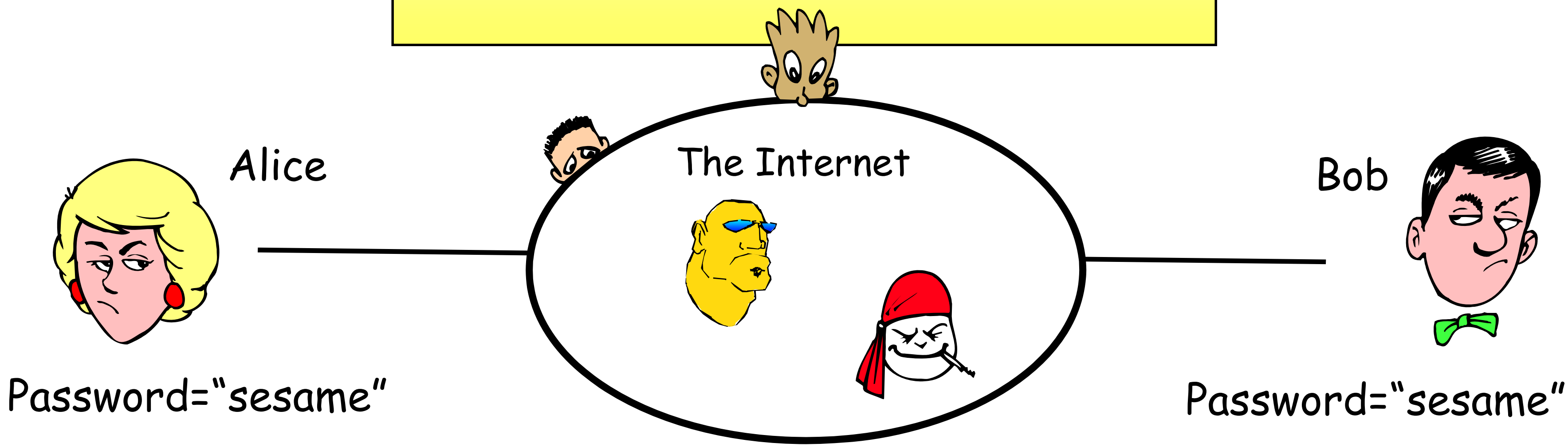


To communicate securely over a network, you need to:

1. Be sure that you're actually talking to the right person.
2. Agree on a crypto key to encrypt / authenticate further communication

Using Passwords

In practice, passwords are used to help attain "secure" channel.



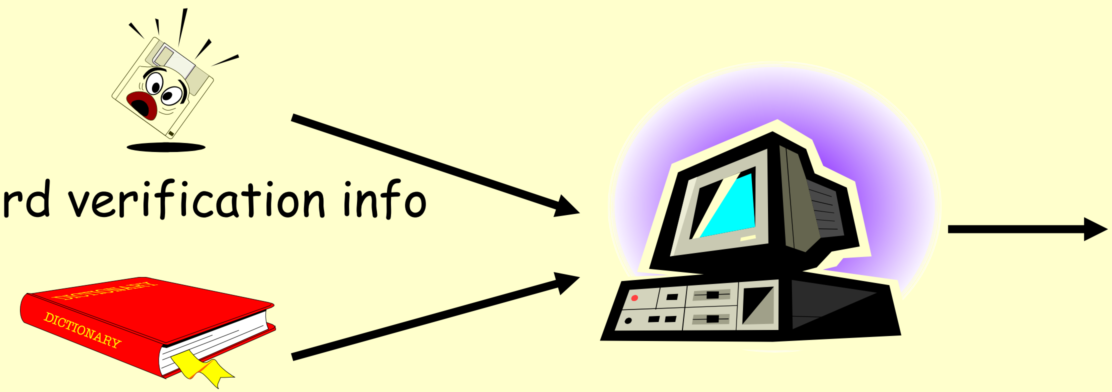
To be sure you're talking to right person, need common secret; e.g., password.

Problem:

Passwords do not have sufficient entropy to be used as cryptographic keys

The Problem with Passwords

Offline dictionary attacks



Example of Offline Dictionary Attack

- Suppose attacker obtains [username, f(password)] pairs, where f is one-way (e.g., a hash function)
- These can be used to verify password guesses!

Dictionary attacks are often very fast and effective.

Classic Password Authentication

In the clear: $\xrightarrow{\text{pw}}$

Password!

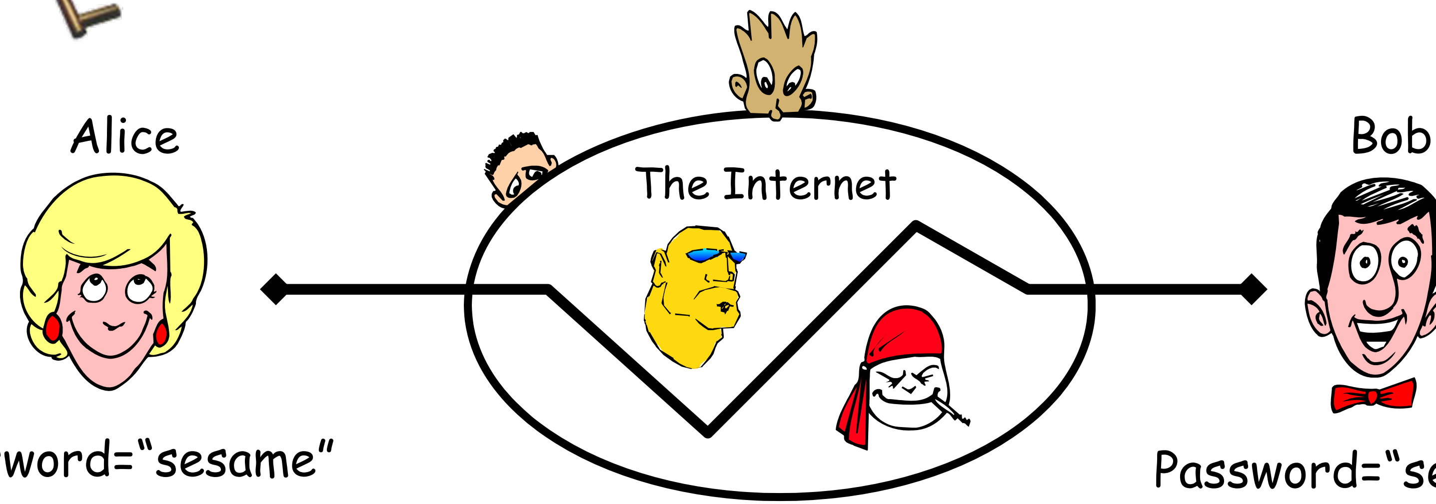
Challenge-response: $\xleftarrow{\text{chal}}$ $\xrightarrow{\text{f(chal,pw)}}$

Password verif. info

Kerberos: $\xrightarrow{\text{req}}$ $\xleftarrow{E_{pw}\text{(vers,key,...)}}$

SSL/ssh: Requires PKI
(or known public keys)

Password Authenticated Key Exchange (PAKE)



Password="sesame"

K=3A82019B7CE8F1F9

Password="sesame"

K=3A82019B7CE8F1F9

PAKE Problem:

Use password for authentication…

And to "bootstrap" a high-entropy session key…

Without using PKI…

While defeating offline attacks.

Attacker's options: eavesdrop, or pose as Alice or Bob

Defenses:

- Online attacks are easy to detect
- *A good PAKE scheme makes offline (dictionary) attacks infeasible*

Prior Techniques

Two prior general approaches for designing efficient PAKE:

- Use password to "encrypt" messages passed in a traditional key exchange protocol (e.g., Diffie-Hellman). Simplified example:

$$\xleftarrow{\quad E_{pw}(g^a) \quad} \qquad \xrightarrow{\quad E_{pw}(g^b) \quad}$$

Crypto key is g^{ab} (in group G)

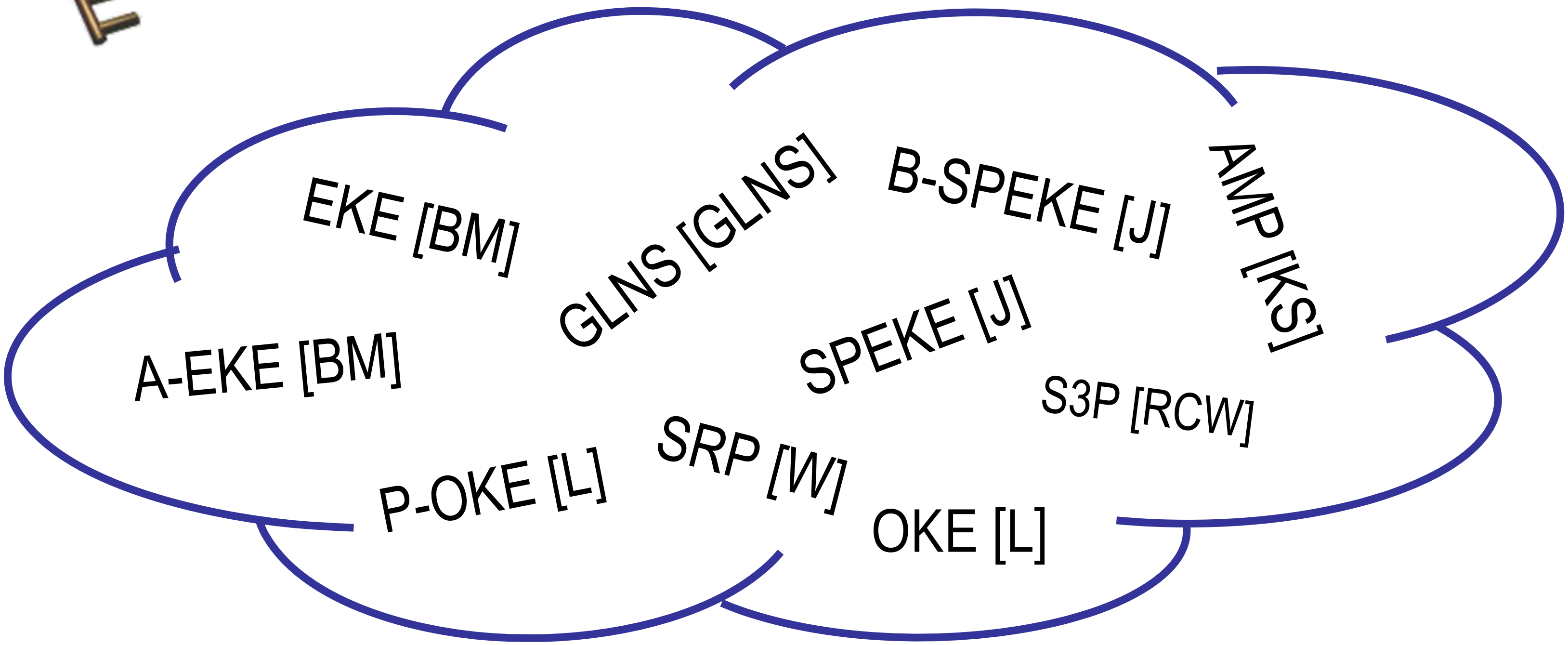
- Use password to generate parameter(s) for a traditional key exchange protocol. Simplified example:

$$\xleftarrow{\quad h(pw)^a \quad} \qquad \xrightarrow{\quad h(pw)^b \quad}$$

Crypto key is $h(pw)^{ab}$ (in group G)

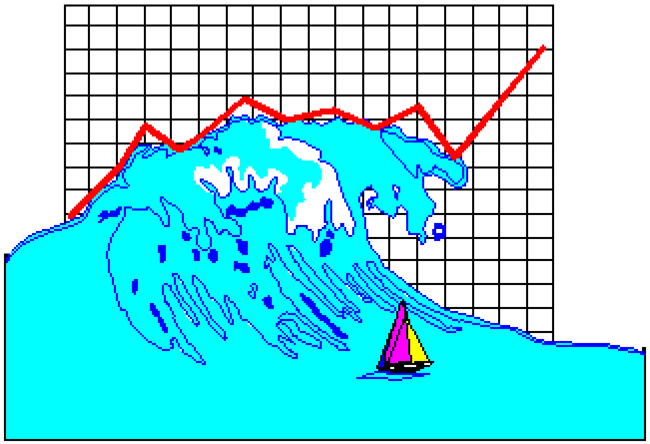
All the (efficient and provably secure) schemes in nearly the last ten years use one of the above techniques… Not much design room left!

The first attempts…



- Problem initially posed by Bellovin and Merritt
- None of the initial protocols came with any proof of security
- In fact, many early versions of some of these protocols were broken subsequent to their publication

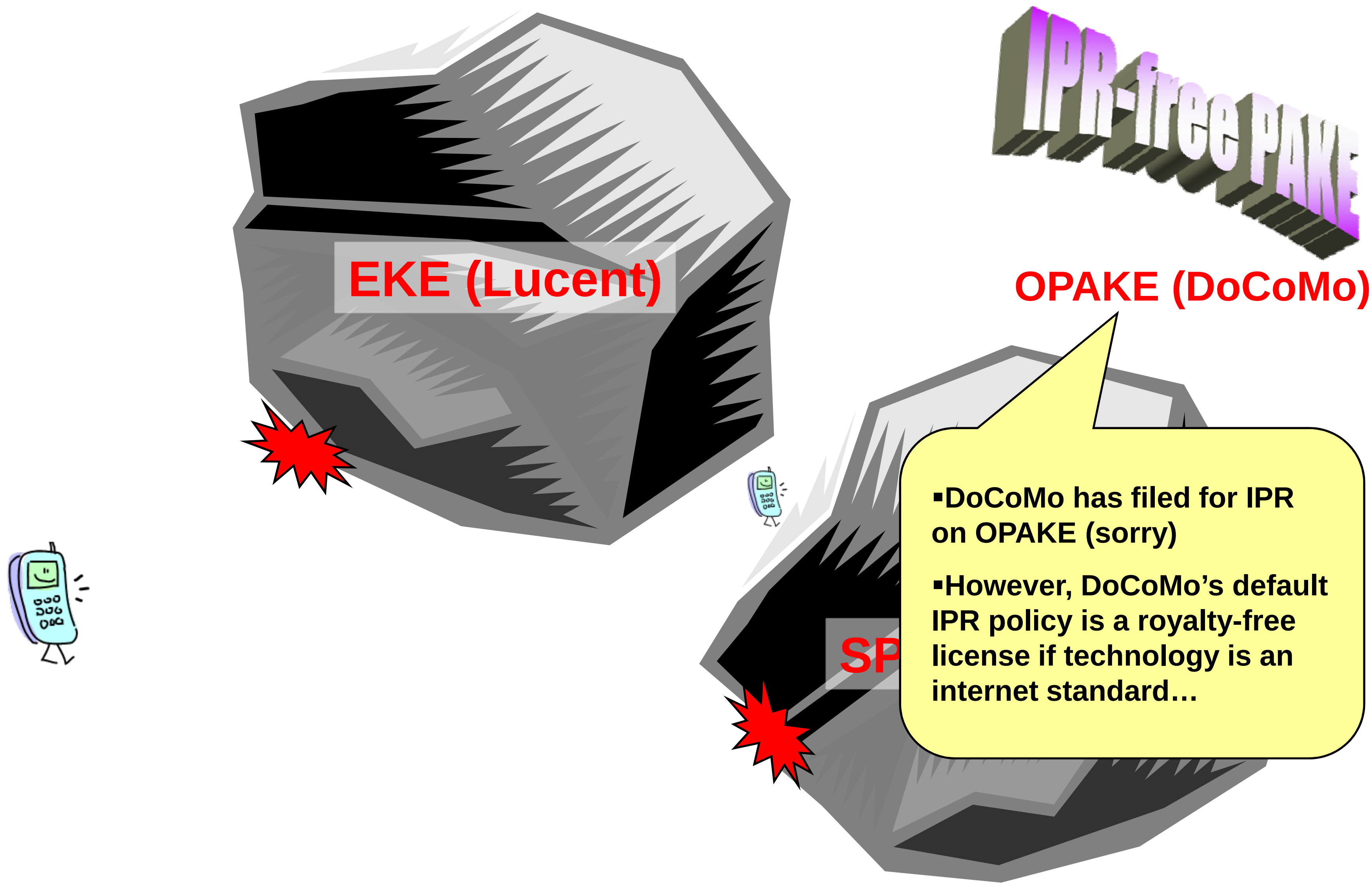
The next wave…



[NP99], [BMP00], [MPS00], [KOY01], [GL01], [GL03], [CPP04], etc.

- Next wave of protocols:
  - Provably secure, based on standard cryptographic assumptions
  - Borrow high-level techniques from earlier protocols
- Outside of [NP99, GL01] these protocols are all fairly efficient.

PAKE patent picture

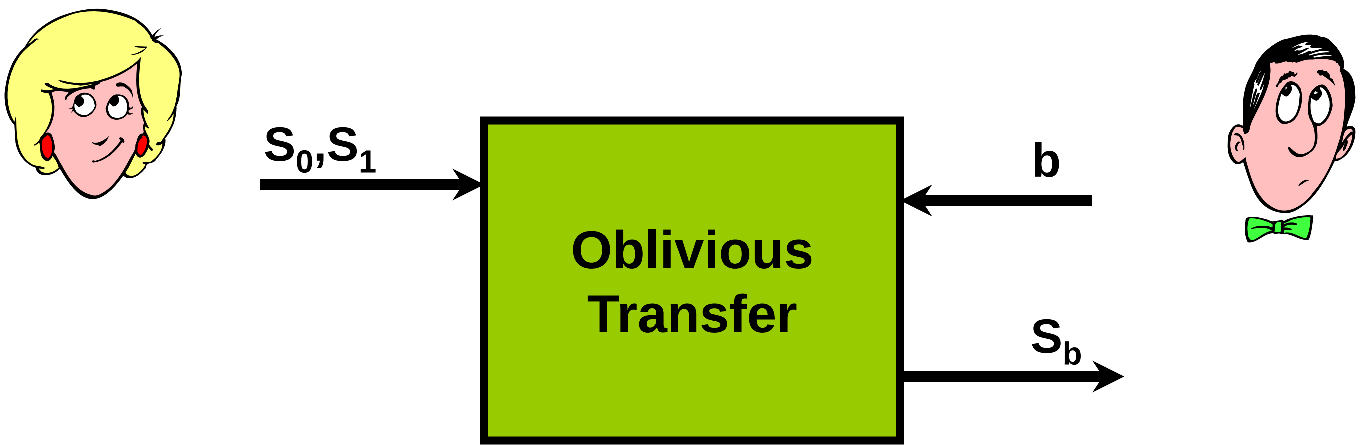


Our Results

- New approach to efficient and provably secure PAKE
- Uses theoretical building block related to Oblivious Transfer (OT).
- But, we use number-theory and algorithmic tricks to do a series of these transfers efficiently "in one shot"
- We have some preliminary implementation results.

Oblivious Transfer (OT)

A fundamental cryptographic technique [Rabin 1981]



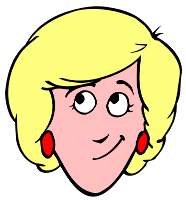
<u>The Setup</u>:

- ✓ Alice has two strings
- ✓ Bob can "choose" to see exactly one of them

- ✓ Sender privacy: Bob learns nothing about the unchosen string
- ✓ Chooser privacy: Alice learns nothing about Bob's choice

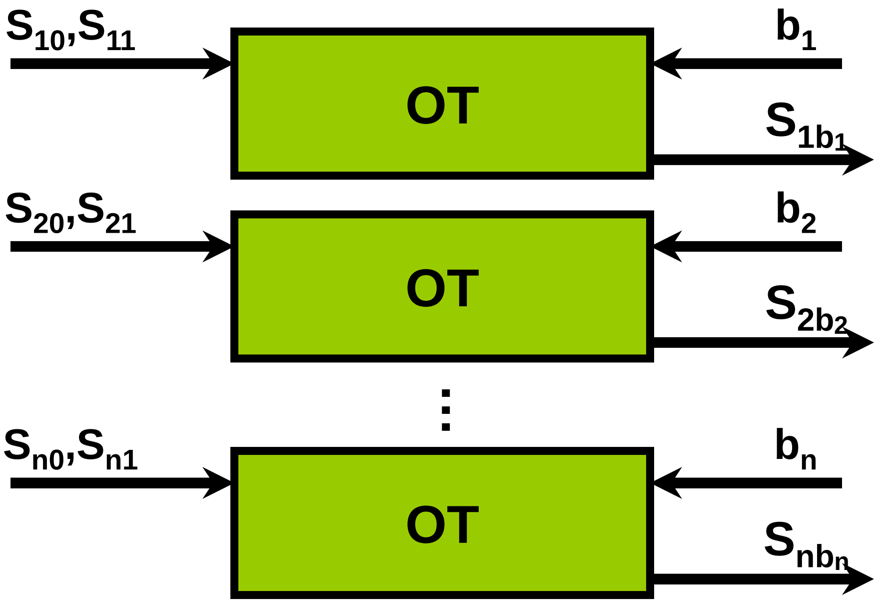
PAKE Using Oblivious Transfer

$pw=b_1b_2\ldots b_n$

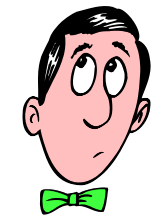


Choose random 80-bit strings:

S_{10},S_{11}
S_{20},S_{21}
:
S_{n0},S_{n1}



$pw=b_1b_2\ldots b_n$



N_A, Hash("Alice", N_A, pw, S_{1b_1}, S_{2b_2}, … , S_{nb_n}) →

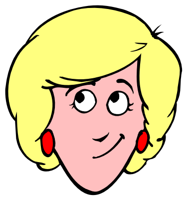
← N_B, Hash("Bob", N_B, S_{1b_1}, S_{2b_2}, … , S_{nb_n})

N_A and N_B are nonces

Crypto key = Hash("Alice", "Bob", N_A, N_B, pw, S_{1b_1}, S_{2b_2}, … , S_{nb_n})

On the OT PAKE Scheme

pw=$b_1b_2 \ldots b_n$

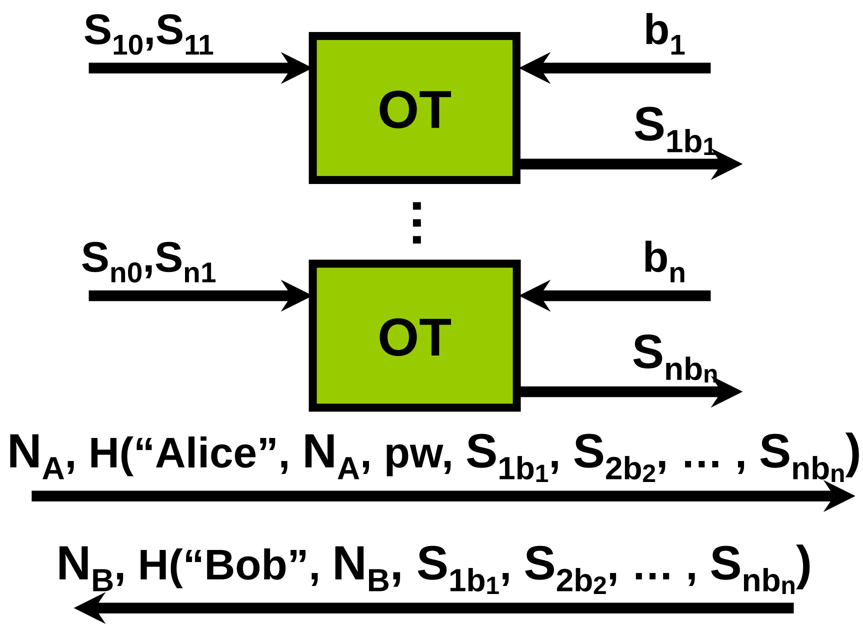


Choose random 80-bit strings:

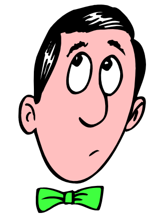
S_{10}, S_{11}

:

S_{n0}, S_{n1}



pw=$b_1b_2 \ldots b_n$



This scheme is slow… There's hope:

OT is very powerful; we really don't need the full benefits of it…

We only need "dumbed-down" OT functionality

- Our strings need not be anything specific, as long as they have sufficient entropy.
- Our scheme will "batch" the OTs efficiently. Also, our OTs need not achieve high security individually, as long as they do in the aggregate.

Two "Dual" Problems

The following two problems are "duals" of each other.

Discrete Logarithm (variant)

Given g and h in group G and prime p, find $DL_g(h)$ mod p. I.e., find (e mod p) where $g^e = h$.

If p divides the order of G:

- DL problem has unique solution.
- RE problem has p solutions.
  (i.e., gx where x is a p^{th} root of unity)

Root Extraction

Given h in group G and prime p, find $h^{1/p}$. I.e., find g where $g^p = h$.

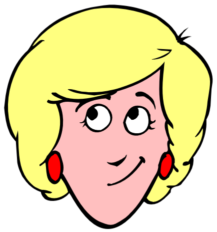
If p doesn't divide the order of G:

- DL problem has p solutions.
  (i.e., $e = e' + r|G| \bmod p$ for integers r and any e' with $g^{e'} = h$).
- RE problem has unique solution.

A First Draft of a PAKE Protocol

Primes associated with each bit of pw

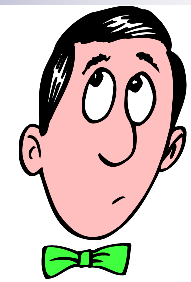
0	1
$p_{1,0}$	$p_{1,1}$
$p_{2,0}$	$p_{2,1}$
⋮	⋮
$p_{n,0}$	$p_{n,1}$



For each i, send two challenges: DL_i & RE_i.

- DL_i is a DL problem: Send g^{e_i} for some e_i. Answer: $e_i \bmod p_{ib_i}$.
- RE_i is a RE problem: Send $g_i^{p_{ib_{i-1}}}$. Answer: g_i.

Finally, compute the hash v=H("Alice",N_A,pw,{e_i},{g_i})



Construct group G such that, for all i, p_{i,b_i} divides |G| and and $p_{ib_{i-1}}$ doesn't.

G, g ←

{DL_i},{RE_i} N_A, v →

Compute {e_i} and {g_i}. Confirm v is correct. w=H("Bob",N_B,{e_i},{g_i}).

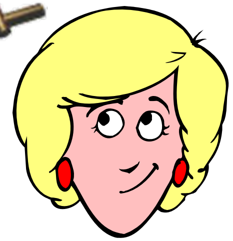
N_B, w ←

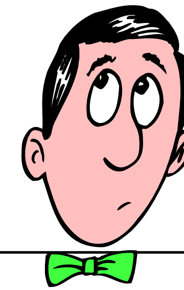
Each challenge effectively contributes log p bits of entropy to the key.

Instantiating the Draft Protocol Efficiently

- Instantiate G as Z_M^* for composite modulus M
  - Decision subgroup problem: Given M, decide if p_{ij} divides $\Phi(M)$
  - DSP is hard → Alice cannot perform offline dictionary attack
- Map passwords to codewords of error-correcting code
  - Different passwords differ at multiple primes
  - Allows us to use smaller primes
- Batch all DL and RE challenges into 2 concise challenges
  - RE challenge solved by a single root extraction
  - DL challenge solved by recursively reducing it to DL problems for small primes, and then using baby-step-giant-step

Our PAKE Protocol





Construct M such that:
$D \mid \Phi(M)$, but $(R, \Phi(M))=1$.
Pick x of order divisible by D.
Note: P = DR = product of primes

M, x ←

Pick random e from $\{0, \ldots MP2^k\}$
a <- e mod D;
y <- x^e mod M
pick random b':
b <- $(b')^D$ mod M;
z <- b^R mod M

v=Hash(1, pw, Bob, trans, a, b)

y, z, v →

x' <- x^(Φ(N)/D) mod M
y' <- y^(Φ(N)/D) mod M;
a <- $\log_{x'} y'$ mod D
b <- $z^{1/R}$ mod M
Verify v

key=Hash(2, pw, trans, a, b)

key=Hash(2, pw, trans, a, b)

Does It Avoid Previous IPR?

Like a DH message or a public key

protocol message

password

- EKE representative claim:

A method for generating a cryptographic key to a first symmetric key cryptosystem by using an authentication signal, said authentication symbol being available to a plurality of parties, said method comprising the steps of:

> forming an outgoing signal by encrypting at least a portion of an excitation signal with a second symmetric key cryptosystem using a key based on said authentication signal, …

e.g., $E_{pw}(g^a)$

We don't encrypt anything symmetrically with a pw-based key.

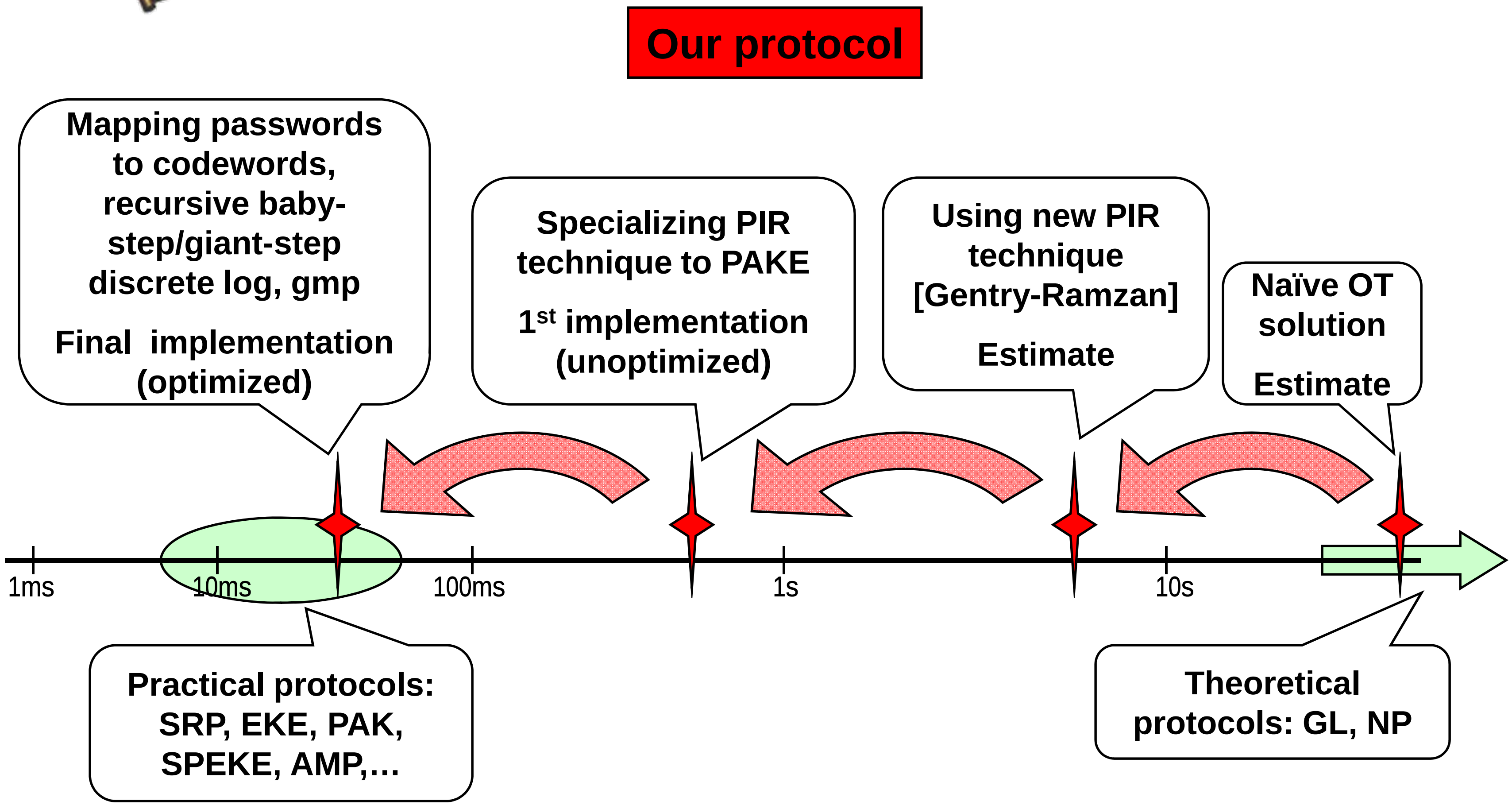
- SPEKE representative claim:

A method… including the steps of: choosing at least one parameter based on said secret value S that defines one of a multitude of means for unauthenticated public key distribution…

e.g., $h(pw)^a$

Claim cannot be valid if read too broadly. Must be limited to specified key distribution schemes, like Diffie-Hellman.

PAKE Efficiency Picture



Performance

Platform	Client (msec)	Server (msec)
Xeon 3.20 GHz	*26*	*23*
Pentium M 2.00 GHz	*46*	*45*
Celeron 2.66 GHz	*100*	*86*

Coded in C, using GMP for multi-precision arithmetic and OpenSSL for basic crypto.

1536-bit modulus; 10-bit primes; (32, 16, 8) ECC.

25 trial runs were averaged; used gettimeofday() for measurements.

Summary

Password Authenticated Key Exchange:

A fundamental tool for providing secure communication

OPAKE: An efficient scheme based on a new approach

Questions?

My email address: cgentry@cs.stanford.edu

Paper in ACM CCS 2005. Email me if you'd like a copy.

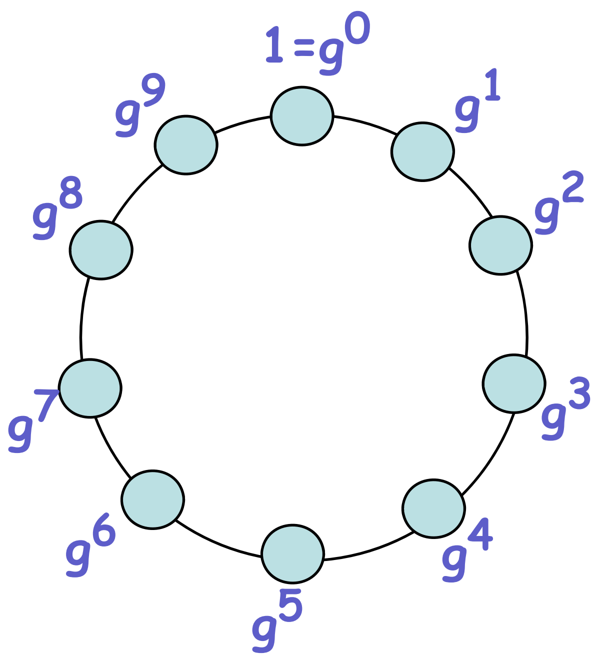
Appendices

High-Level Description of Our Approach

- **Server (Bob) uses password to generate group G**: Group order divisible by of one set of primes, but not another (based on password); description, generator sent to Client (Alice).
- **Client (Alice) "challenges" Server (Bob)** with discrete log and root extraction instances in G. Problem instances set up based on what Alice thinks factorization of the group order should be (based on password).
- **Server (Bob) solves instances**: Instances have unique solutions only if G chosen correctly.
- **Hash everything**: solutions, password, and transcript of communication to generate cryptographic key.

Security intuition: If Bob doesn't know correct password, then he'll be either solving DL instances with the "wrong" subgroup order or taking the "wrong" root (or both!). Result: many solutions to problem instances.

Root Extraction in Cyclic Groups



Imagine a cyclic group of order 10; now, 3 does not divide group order, but 5 does.

Every element has a unique "3rd root":

3, 6, 9, 12, 15, 18, 21, 24, 27, 30

(mod 10)

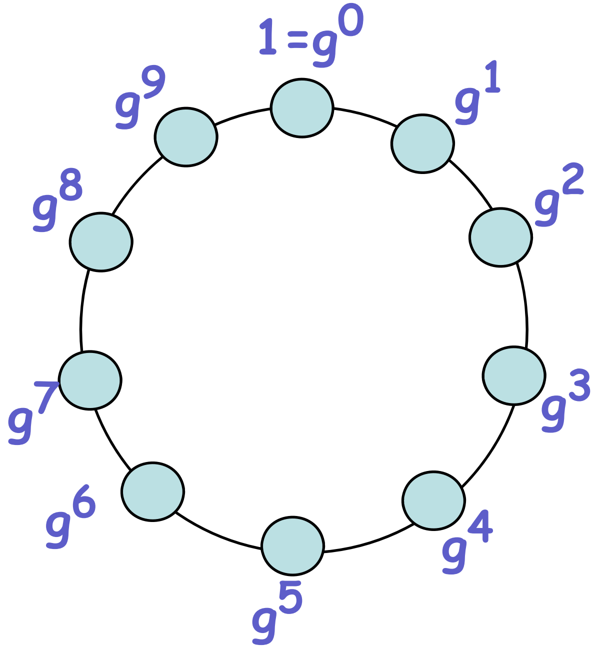
3 6 9 2 5 8 1 4 7 0

But, there is no *unique* "5th root"

e.g., the 5th roots of 5 are 1, 3, 5, 7, 9:

5 = 1*5 = 3*5 = 7*5 = 9*5 mod 10

Discrete Logs in Cyclic Groups



Imagine a cyclic group of order 10; now, 3 does not divide group order, but 5 does.

Consider group generated by "2": 2, 4, 6, 8, 0. Every element has a unique discrete logarithm modulo 5 (but not mod 3):

log 2 = 1 (mod 5);
log 4 = 2 (mod 5);
log 6 = 3 (mod 5);
log 8 = 4 (mod 5);
log 0 = 0 (mod 5).

log 2 = 1 (mod 3);
log 4 = 2 (mod 3);
log 6 = 0 (mod 3);
log 8 = 1 (mod 3);
log 0 = 2 (mod 3).