

# Secure Authentication without (Traditional) Cryptographic Keys

# Motivation

"Humans are incapable of securely storing high-quality cryptographic secrets, and they have unacceptable speed and accuracy.... (They are also large [and] expensive to maintain.... But they are sufficiently pervasive that we must design our protocols around their limitations.)"

From: "*Network Security: Private Communication in a Public World*," by Kaufman, Perlman, and Speciner

# Take-away point

- Humans simply cannot memorize high-entropy cryptographic keys
  - Token-based solutions possible, but often inconvenient/expensive
- “Traditional” crypto cannot be used
  - Symmetric key, public key...
- What other options are there?

# Possibilities...

- Passwords (PAK protocols)
  - Low-entropy, non-uniform strings
- Biometric data
  - High-entropy data "for free"
- Passwords probably more convenient, but achievable security is limited

# Password-based authentication (and key exchange)

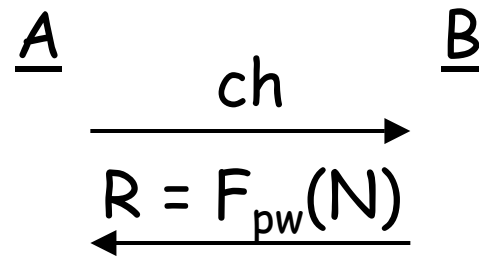
# Some terminology

- Cryptographic **keys** (high entropy)
  - Exhaustive search infeasible (e.g., 128 bits)
  - Secure when used to encrypt / authenticate communication
- **Passwords** (low entropy)
  - Typically chosen from a known dictionary
  - Exhaustive on-line attacks feasible!
  - *Insecure* when used to encrypt / authenticate communication

# Remark

- A protocol may be secure when using **keys**, but completely insecure when using **passwords**!
- Example...

# Challenge-response



- After eavesdropping on a **single** protocol execution, adversary can try all passwords in the dictionary to match  $F_*(ch)$  to  $R$
- Example of an **off-line dictionary attack**

# Basic goal

- Construct a protocol where an **on-line** dictionary attack is the *only* attack
  - In particular, off-line dictionary attacks should not help at all

# Remember this slide

- The problem boils down to constructing a protocol in which each on-line attack by an adversary corresponds to a **single** password guess
  - (Taking into account man-in-the-middle attacks as well as off-line attacks)

# My work in this area...

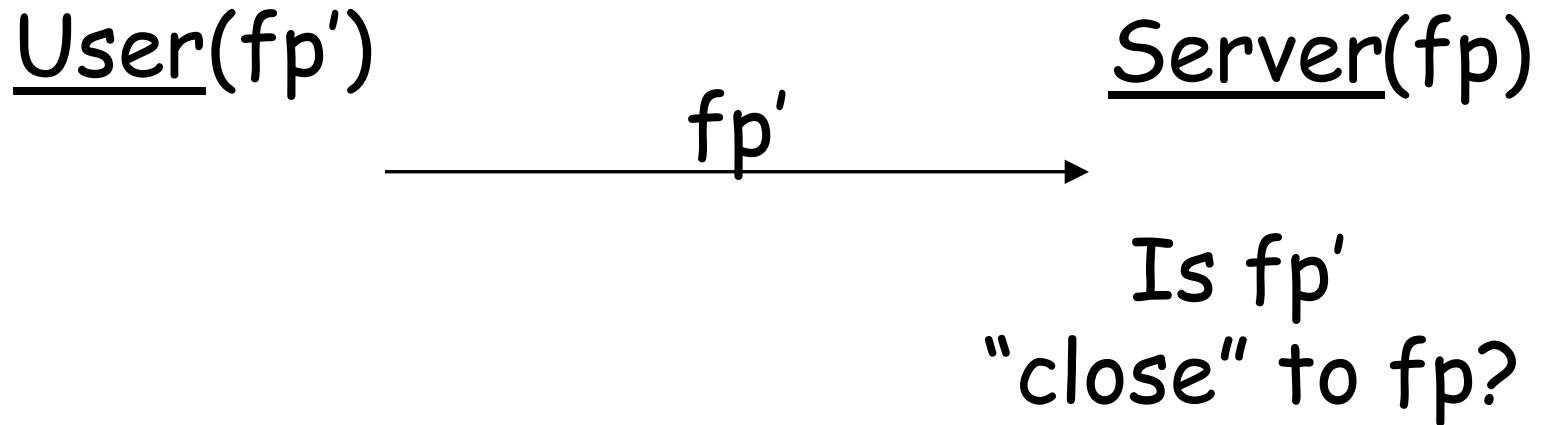
- First provably-secure and efficient solution (Eurocrypt '01)
- Stronger definitions of security, and efficient protocols w.r.t. this definition (Eurocrypt '05)
- Improved efficiency; two-server solutions (ACNS '05)

Using biometric data for  
secure remote authentication  
(Eurocrypt '05)

# Problems with biometrics

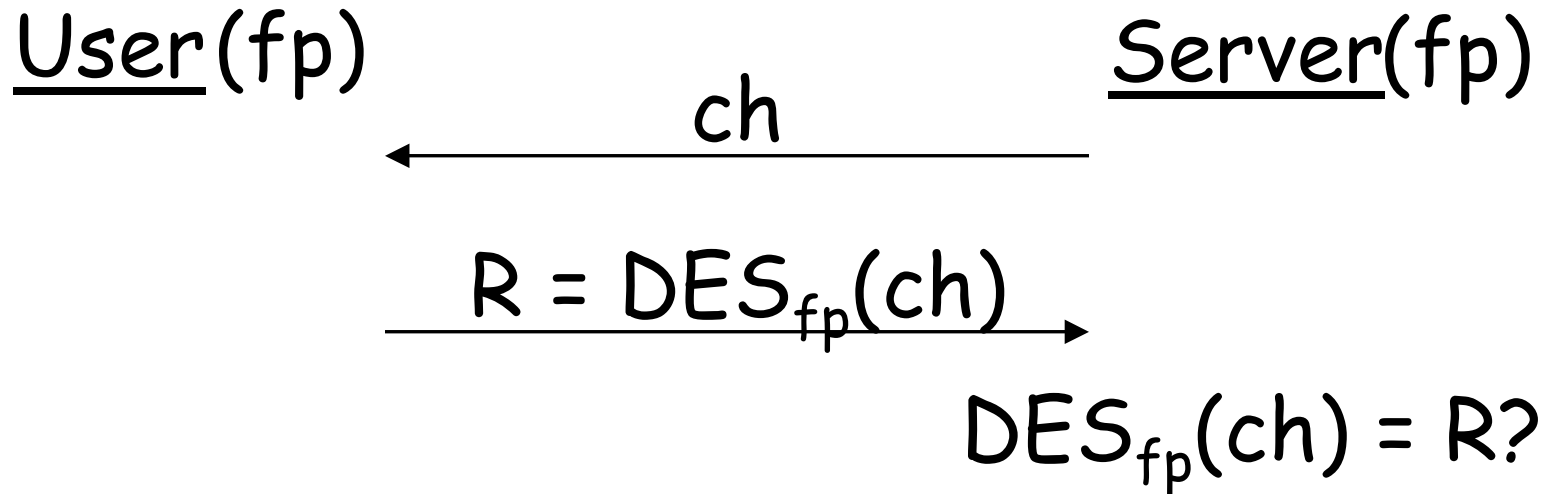
- At least two important issues:
  - Biometrics are not **uniformly random**
  - Biometrics are not **exactly reproducible**
- Outside the scope of this talk
  - Are biometrics private?
  - Sufficiently-high entropy?
  - Revocation?

# Some ideas...?



Insecure against a passive eavesdropper!

# Some ideas...?



Who says fingerprints make good DES keys...?



# What do we want?

- Let's formalize the problem...
- Ultimately, want authentication secure against **active** adversaries
- Break it down into simpler problems first:
  - Secure sketches
  - Fuzzy (randomness) extractors

# One piece of notation

- The min-entropy of a random variable is the negative logarithm of the probability of the **most likely value**
  - If  $X$  has min-entropy  $k$ ,  $X$  cannot be predicted with probability  $>2^{-k}$

# Modeling biometric data

- View biometric readings as forming a discrete **metric space**  $(M, d)$ 
  - For simplicity, use Hamming metric
- Errors: adversary specifies **arbitrary** sequence of random variables  $(W_0, W_1, \dots)$  such that  $d(W_0(\omega), W_i(\omega)) \leq \tau$  for all  $i$

# Secure sketch

- Want a way for the user to be able to **recover** their "true" biometric data  $w$  from a "close" copy  $w'$ ...
- But should be infeasible for an adversary to recover  $w$  without  $w'$ 
  - In particular, the recovery procedure should not leak "too much" about  $w$

# Formal definition

- $(m, m', t)$ -secure sketch  $(SS, Rec)$ :
  - For all  $w, w'$  with  $d(w, w') \leq t$ :  
 $Rec(w', SS(w)) = w$   
(I.e., "recovery from error")
  - If  $W$  has min-entropy  $\geq m$ , the average min-entropy of  $W$  given  $SS(W)$  is  $\geq m'$   
(I.e., "w still hard to guess")

# Fuzzy extractor

- Recovered  $w$  is still not **uniformly** distributed!
  - Apply randomness extraction to any secure sketch...
  - ...using, e.g., pair-wise independent hashing



# Constructing a secure sketch

- Let  $(E,D)$  be a error-correcting code
  - Correcting  $t$  errors
- To generate a sketch of  $w$ , choose random  $m$  and set  $SS(w) = E(m) \oplus w$
- To recover  $w$  given  $SS(w)$  and  $w'$ , do:
  - (1)  $m = D(SS(w) \oplus w')$
  - (2)  $w = E(m) \oplus SS(w)$

# Analysis

- Correctness is immediate
- Security:
  - $(m,w)$  together have min-entropy  $|m| + k$
  - $SS(w)$  reduces min-entropy of  $(m,w)$  by at most  $|SS(w)|$
  - Given  $SS(w)$ ,  $w$  uniquely determines  $m$
  - Min-entropy loss of  $w$  is at most  $|SS(w)| - |m|$

# Active adversaries

- Two approaches (so far):
  - *Robust* extractors
  - Using PAK protocols
    - These are designed for use with "short" passwords...
    - ...but no reason to limit their use to low-entropy secrets!

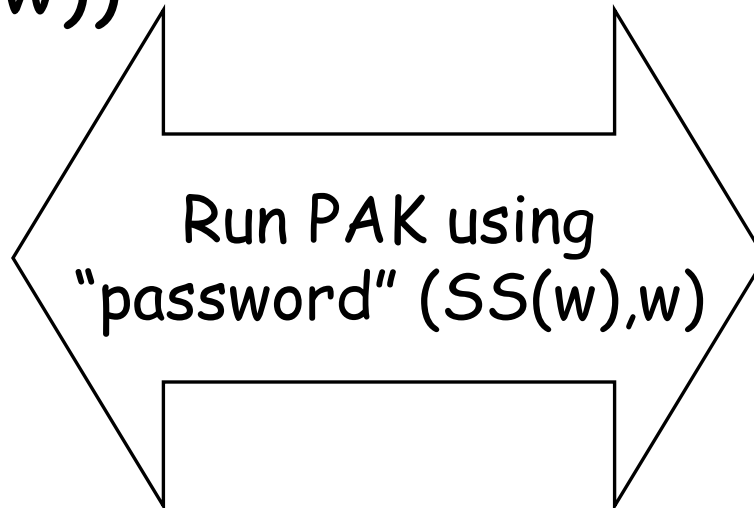
# Application to remote authentication

User ( $w'$ )

Server ( $w$ )

$SS(w)$

$w = \text{Rec}(w', SS(w))$



# Intuition

- Even if adversary changes  $SS(w)$ , the value  $w'$  recovered by the user still has "high enough" min-entropy
- By security of PAK protocol, adversary reduced to guessing this  $w'$

# Conclusions

- Research motivated by limitations of human users
  - I.e., inability to remember high-entropy keys
- Questions?