User Authentication : Passwords, Biometrics and Alternatives

Pierre-Alain Fouque

Introduction

- Username and passwords to access local device or remote account
- Authentication: The process of using supporting evidence to corroborate an asserted identity
- Identification (recognition) establishes an identity from available information without an explicit identity having been asserted
  - picking out known criminals in a crowd or finding who matches a given fingerprint; each crowd face is checked against a list of database faces
- For identification, since the test is one-to-many, problem complexity grows with the number of potential candidates
- Authentication involves a simpler one-to-one test

Introduction

- Corroborating an asserted identity may be an end-goal or a sub-goal towards the end-goal authorization (determining whether a requested privilege or resource access should be granted to the requesting entity)
  - E.g.: users may be asked to enter a password (for the account currently in use) to authorize installation or upgrading of O.S. or application software
- We are interested here on user authentications – humans being authenticated by a computer system
  - Our main topic are passwords, hardware-based tokens, and biometric authentications

Password authentication

- Passwords provide basic user authentication
  - each user authorized to use a system is assigned an account identified by a character string username (or numeric userid)
- To gain access (``log in'') to their account, the user enters the username and a password
- This pair is transmitted to the system, which has stored information to test whether the password matches the one expected for that userid. If so, access is granted.
  - A correct password does not ensure that whoever entered it is the authorized user. That would require a guarantee that no one other than the authorized user could ever possibly know, obtain, or guess the password– which is unrealistic. A correct match indicates knowledge of a fixed character string—or possibly a ``lucky guess''
  - But passwords remain useful as a (weak) means of authentication.

Storing hashes vs. cleartext

- To verify entered userid-password pairs, the system stores sufficient information in a password file F with one row for each userid
- Storing cleartext passwords p_i in F would risk directly exposing all p_i if F were stolen; system administrators and other insiders, including those able to access filesystem backups, would also have all passwords
- Instead, each row of F stores a pair (userid,h_i), where h_i=H(p_i) is a password hash; H is a publicly one-way hash function. The system computed h_i from the user-entered p_i to test for a match

Pre-computed dictionary attack

If password hashing alone is used as above, an attacker can perform a pre-computed dictionary attack:

1. Construct a long list of candidate passwords w_1, …, w_t
2. For each w_j, compute h_j=H(w_j) and store a table T of (h_j,w_j) stored by h_j
3. Steal the password file F containing stored values h_i=H(p_i)
4. ``Look up'' the password p_i corresponding to a specified targeted userid u_i with password hash h_i by checking whether h_i appears in T as any value h_j; if so, w_j works as p_i. If instead the goal is to trawl (find passwords for arbitrary userids), sort F's rows by values h_i, then compare sorted tables F and T for matching hashes h_j and h_i representing H(w_j) and H(p_i); this may yield many matching pairs and w_j will work as u_i's password p_i

Targeted vs. Trawling scope

- The pre-computed attack considered:
  - A targeted attack specifically aimed at pre-identified users (often one);
  - A password trawling attack aiming to break into any account by trying many or all accounts

Approaches to defeat password authentication

- Password authentication can be defeated by several approaches:

1. Online password guessing. Guesses are sent to the legitimate servers
2. Offline password guessing. No per-guess online interaction is needed
3. Password capture attacks. An attacker intercepts or observes passwords: observing sticky-notes, shoulder-surfing, server-side interception, proxy or middle-person attacks, phishing and social engineering, hardware or software keylogger, ….
4. Password interface bypass: aim to defeat authentication mechanisms by gaining unauthorized access by exploiting software vulnerabilities
5. Defeating recovery mechanisms

Password composition policies and strength

- To ease the burden of remembering passwords, many users choose (rather than strings of random characters) words found in common-language dictionaries
- Since guessing attacks exploit this, many system impose password composition policies with rules specifying minimum lengths (8 or 10) and requiring password characters (LUDS) Lowercase, Uppercase, Digits and Special characters
- Such passwords are said to be stronger but this term misleads in that such increased complexity provides no more protection against capture attacks

Disadvantages of passwords

- Usability challenges multiply as the number of passwords that users must manage grows from just a few to tens of hundreds
- Usability disadvantages include users being told:
  - Not to write their passwords down
  - To follow complex composition policies
  - Not to re-use passwords across accounts
  - To choose each password to be easy remember but difficult for others to guess
  - To change passwords every 30—90 days if password expiration policies are in use

Advantages of passwords

- Passwords
  - Are simple, easy to learn, and already understood by all current computer users
  - Are free (requiring no extra hardware at the client or system/server)
  - Require no extra physical device to carry
  - Allow relatively quick login, and password managers may help further (for small keyboard mobile device, apps commonly store passwords);
  - Are easy to change or recover if lost
  - Have well-understood failure modes (forgetful users learn to write passwords down somewhere safe)
  - Require no trust in a new third party (contrary to certificates)
  - Are easily delegated

Password-guessing strategies and defenses

- Online Password guessing and rate-limiting
  - can be mounted against any publicly reachable password-protected server. Userid-password pairs, with password guesses, are submitted sequentially to the legitimate server, which indicates if the attempt is successful or not
  - An obvious defensive tactic, for sites that care at all about security, is to rate-limit or throttle guesses across fixed time windows—enforcing a maximum number of incorrect login attempts per account
- Offline Password guessing
  - Assumed that an attacker has stolen a copy of a system's password hash file
  - On Unix-based systems: /etc/passwd : verifiable text and no rate-limiting

Password-guessing strategies and defenses

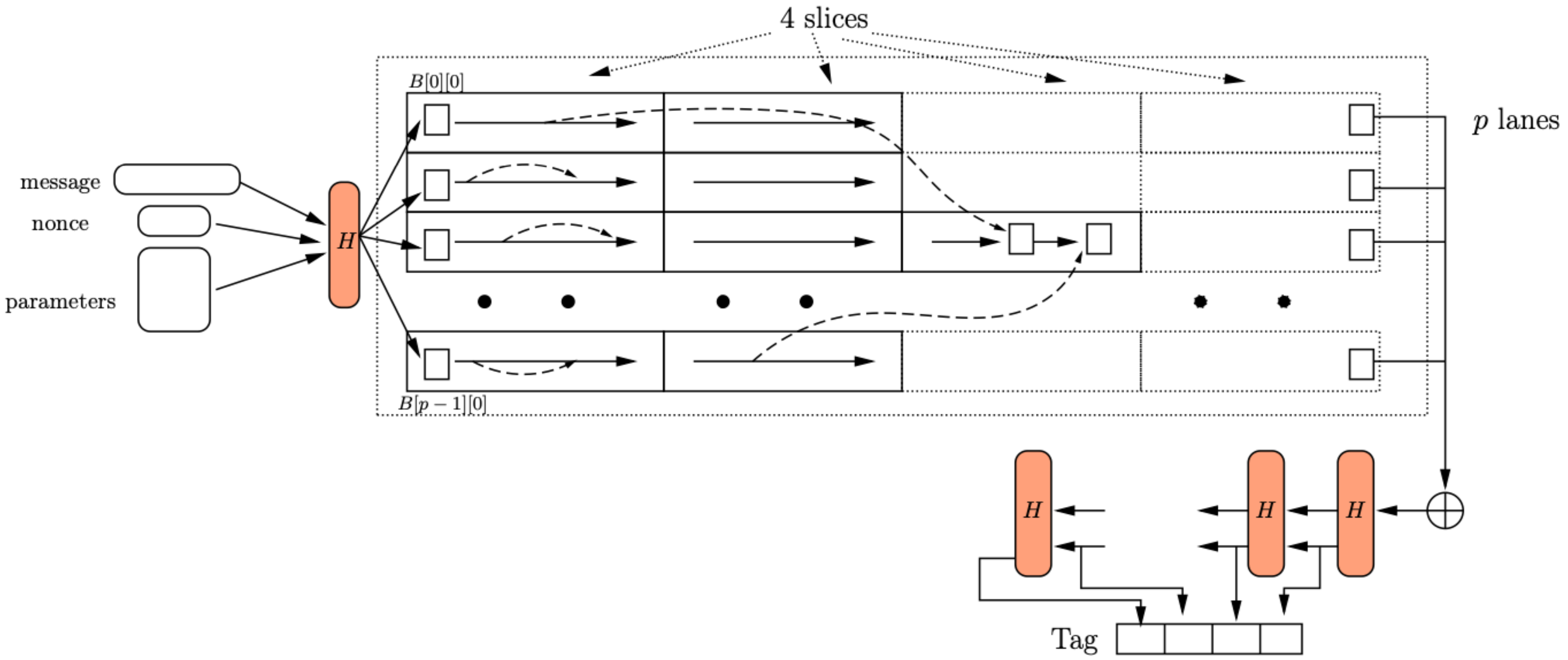
- Iterated Hashing (password stretching)
  - Offline attacks can be slowed down using iterated hashing (password stretching): after hashing a password once with hash function H, rather than storing $H(p_i)$, the result is itself hashed against d-fold $H(\ldots H(H(p_i))\ldots)$. d=1000 slows attacks by a factor 1000
- Password salting
  - To combat dictionary attack, common practice is to salt passwords before hashing. For userid u_i, on registration of each password p_i, rather than storing $h_i=H(p_i)$, the system selects t>=64, a random t-bit value s_i as salt and stores (u_i, s_i, $H(p_i,s_i)$) with p_i, s_i concatenated before hashing
  - The password is altered by the salt in a deterministic way before hashing. For trawling attacks, the work is harder by a factor 2^t in computation and storage
  - Complexity not increased for a targeted userid

Password-guessing strategies and defenses

- A bonus of salting is that two users with the same passwords will almost certainly have different password hashes in the hash file
- Pepper (secret salt)
  - A secret salt (pepper) is like a regular salt but not stored: slow down by a different than the iterated hashing. When user u_i selects a new password p_i, the system chooses a random value $1<=r_i<=R$, stores the secret-salted hash $H(p_i,r_i)$ and then erases r_i. To later verify a password for account u_i, the system sequentially tries all values $r^*=r_i$ in a deterministic order. If R is 20 bits, on average slow-down by a factor 2^{19}. Pepper can be combined with salt …

Password-guessing strategies and defenses

- Specialized password-hashing functions
    - PBKDF2 (password-based KDF number 2), Argon2i, bcrypt, scrypt against efficient hash function with specialized hardware



Password-guessing strategies and defenses

- System-assigned passwords and brute-force guessing
- The difficulty of guessing passwords is maximized by selecting each password character randomly and independently. An n-character password chosen from a b characters then results in b^n possible pwd
- Brute-force guessing: sequential enumeration. The probability of success is 100% after b^n guesses, 50% after $b^n/2$ guesses. If the passwords need not be a full n characters, a common strategy first try all one-character passwords, then all two-character passwords, …

Probability of guessing success

- Maximum guessing probabilities of $1/2^{10}$ (Low) and $1/2^{20}$ (Higher)
- Parameters:
  - G: the number of guesses the attacker can make per time unit
  - T: the number of time units per guessing period
  - $R=b^n$: the size of the password space (equiprobable passwords)
- Assume password guesses can be verified by online or offline attack
- The probability q that the password is guessed by the end of the period is the proportion of the password space an attacker can cover:
  - If GT>R, q=1.0
  - q=GT/R otherwise GT <= R

Example: Offline guessing

- $T=1$ year (3.154×10^7 s)
- truly random passwords of length $n=10$ from an alphabet $b=95$ (printable characters): $R=b^n=95^{10}=6 \times 10^{19}$; and $G=100$ billion guesses per second (modest number of modern GPUs)
- $q=GT/R=(10^{11})(3.154 \times 10^7)/6(10^{19})=0.05257$
- Success probability over 5% greater than both 2^{-10} and 2^{-20}
- Lower Bound on length: $n=\lg(R) / \lg(b)$ where $R=GT/q$
- Alternatively to model an online attack, one can determine what degree of rate-limiting suffices for a desired q, from $G=qR/T$

Password distributions

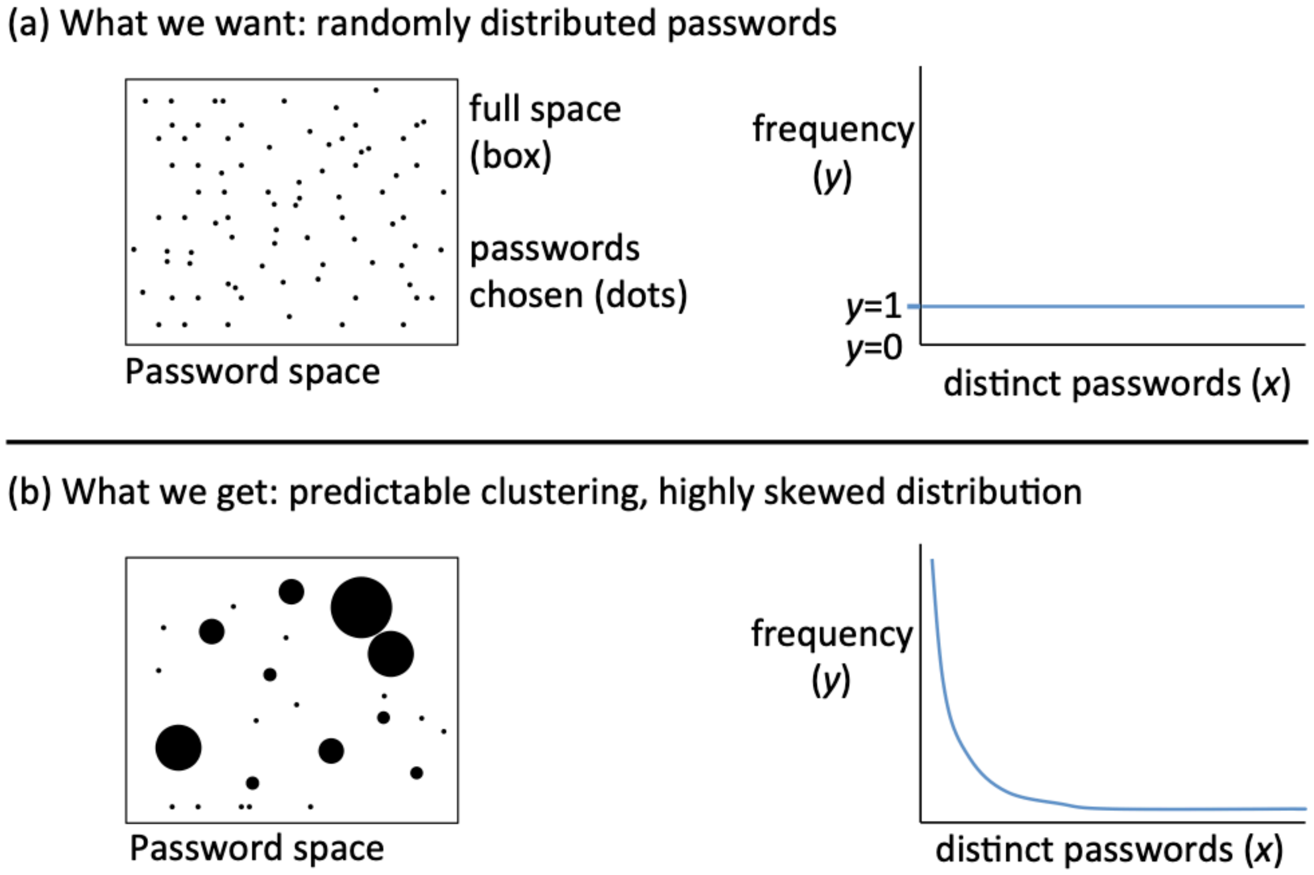


Figure 3.2: Password distributions (illustrative). Ideally, chosen passwords are unique ($y = 1$) with most unchosen ($y = 0$). Diameter represents frequency a password is chosen.

Other remarks

- Heuristic password-cracking tools: JohnTheRipper and oclHashcat
- Login passwords vs. Passkeys : passwords may be used to derive cryptographic keys for file encryption. Such password-derived encryption keys (passkeys) are subject to offline guessing attacks and require high guessing resistance
- Password management: NIST SP 800-63B
- Account recovery and secret questions
- Password managers
- Graphical passwords

One-Time password generators and hardware tokens

OTPs from Lamport hash chains. Starting with a random secret (seed) w, user A can authenticate to server B using a sequence of one-time passwords as follows (Fig. 3.3). H is a one-way hash function (Chapter 2) and t is an integer (e.g., $t = 100$). Let a *hash chain* of order t be the sequence: $w, H(w), H(H(w)), H^3(w), ..., H^t(w)$. H^t means

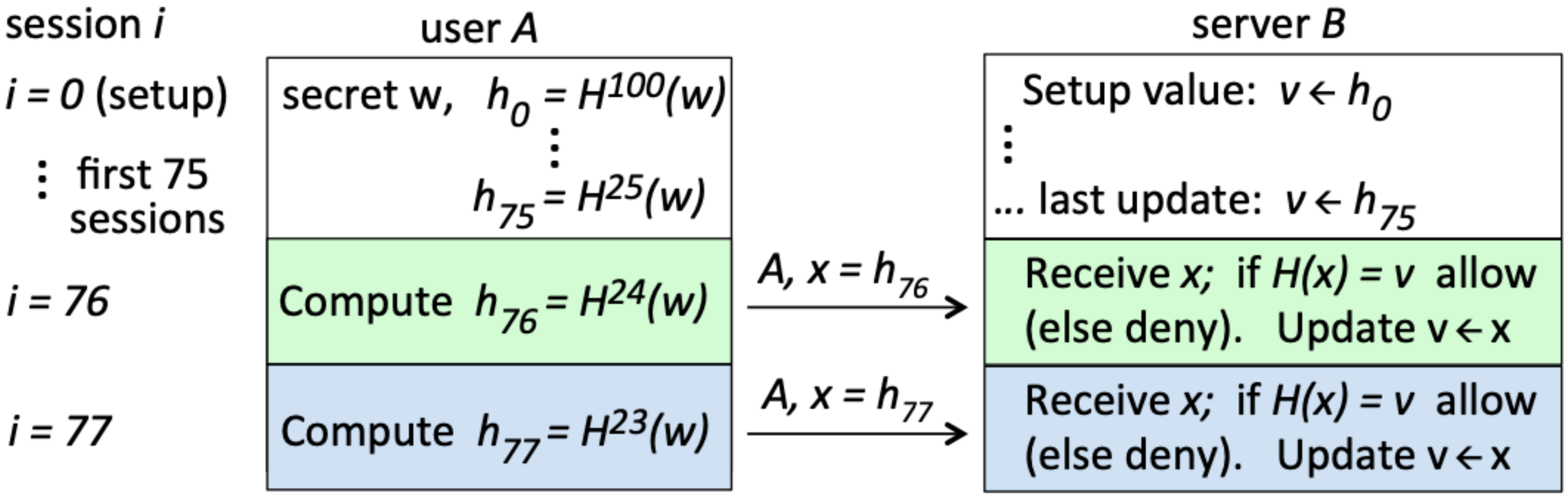


Figure 3.3: Lamport hash chain. Illustrated with $t = 100$ for session $i = 76$ $(t - i = 24)$. Setup value h_0 must initially be transferred over a secure channel and associated with A.

Multiple factors

- Two-factor authentication (2FA) requires the methods be from two different categories

User authentication categories based on type of verification evidence

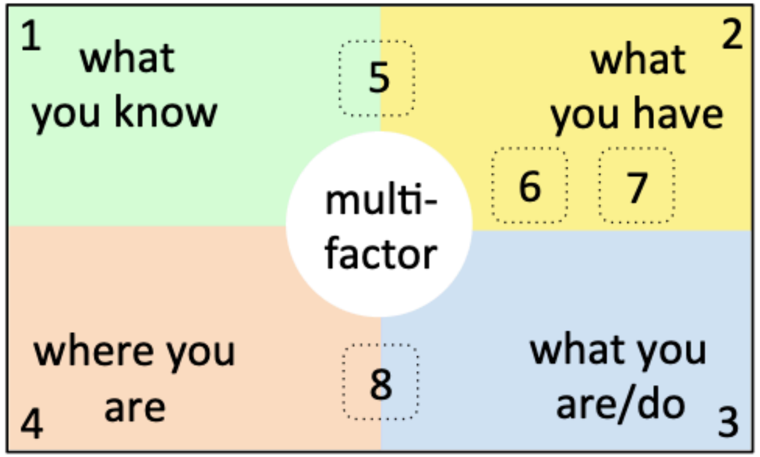


Figure 3.5: User authentication categories 1–3 are best known. Here (3) includes physical and behavioral biometrics; behavioral patterns could be considered a separate category, e.g., observed user location patterns (8). Location-based methods (4) may use *geolocation* of a user-associated device. A secret written on paper (because it is critical yet might be forgotten, or rarely used) may be viewed as something you have (5). Devices may receive one-time passwords (6). Device fingerprinting is shown as a sub-category of (7).

Biometric authentication

- Biometric authentication: physical biometrics (what you are); behavioral biometrics (what you do)
- Biometrics are non-secrets

Modality	Type	Notes
fingerprints	P	common on laptops and smartphones
facial recognition	P	used by some smartphones
iris recognition	P	the part of the eye that a contact lens covers
hand geometry	P	hand length and size, also shape of fingers and palm
retinal scan	P	based on patterns of retinal blood vessels
voice authentication	M	physical-behavioral mix
gait	B	characteristics related to walking
typing rhythm	B	keystroke patterns and timing
mouse patterns	B	also scrolling, swipe patterns on touchscreen devices

Table 3.2: Biometric modalities: examples. P (physical), B (behavioral), M (mixed). Fingerprint (four digits) and iris biometrics are used at U.S.-Canadian airport borders.

Biometric process: enrollment and verification

- Features are extracted from the Reference template
- Matching score s=0 (no similarity) s=100 (100% agreement)
- Threshold t If s>=t, the system declares the sample to be from the same individual
- False rejects: a legitimate user's new sample is declared to not match their own template. False Reject Rate (FRR): FRR=Prob[System declares X_V does not matches X_L | X_V is sampled from L]
- False accept: an imposter's sample is (wrongly) declared to match the legitimate user's template. False Accept Rate (FAR)
    - Fixing a threshold t and legitimate user L with reference template X_L, let X_V denote the biometric samples to be matched.
    - FAR=Prob[X_V matches X_L | X_V is different from L]
- EER: Equal Error Rate: is the point at which FAR = FRR simplified single-point comparisons – the system with lower EER is preferred

Biometric system tradeoff

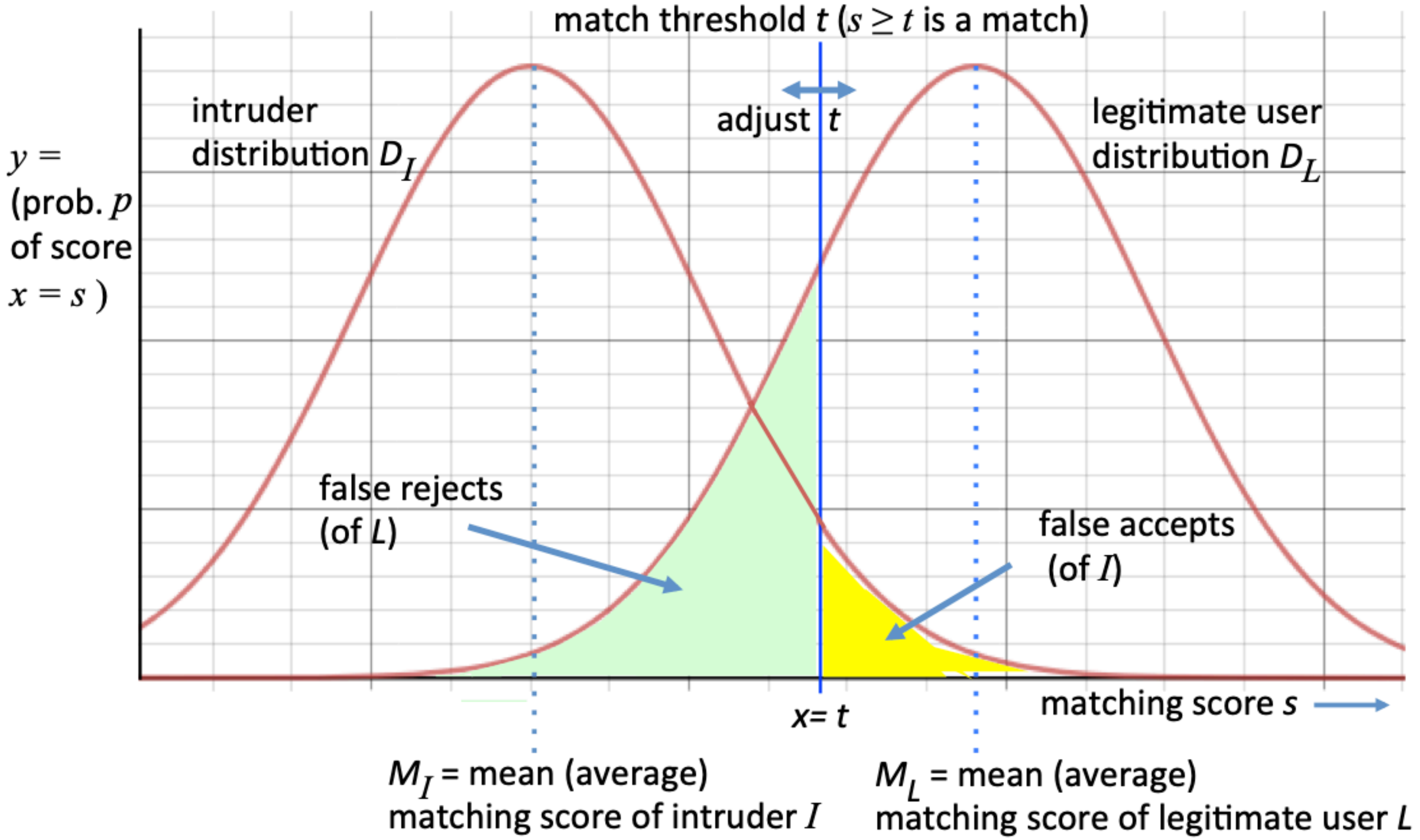


Figure 3.6: Biometric system tradeoffs. Curves model probability distributions for an intruder and legitimate user's matching scores; higher scores match the user's biometric template better. The y axis reflects how many biometric samples get matching score $x = s$.

DET (Detection Error Tradeoff) and ROC Relative/Receiver operating curves

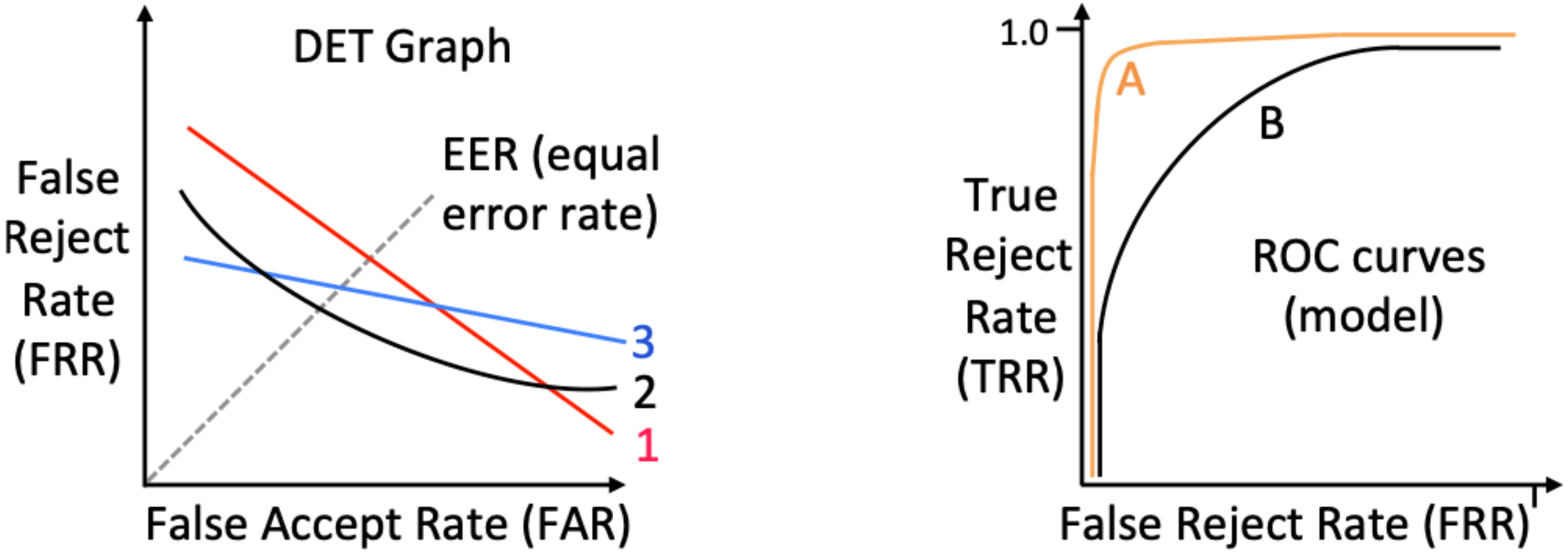


Figure 3.7: DET graph and ROC curve. These depict a system's characteristics for different values of a decision threshold t, and allow comparisons between systems. If EER is used as a single comparison point, System 2 is preferred. System 3's FRR decreases slowly as parameters are adjusted to admit a higher FAR; in contrast, System 1's FRR decreases more rapidly in return for an increased FAR. An upper-left ROC curve is better (A). In binary classification of events in the intrusion detection scenario (Chapter 11), the analogous terminology used is True/False Positive Rate, and True/False Negative Rate.

Entropy, passwords, and partial-guessing metrics

- Data, information representation, and entropy: a 16-bit word might be used to convey four values. The same information can be conveyed in 2 bits. For the given probabilities, in information theory we say that there are 2 bits of entropy

Information	Probability	Hex representation	Binary alternative
red	0.25	`0000`	00
green	0.25	`00FF`	01
blue	0.25	`FF00`	10
black	0.25	`FFFF`	11

Table 3.3: Alternative representations for conveying four known values. The same information is conveyed, whether two bytes of data are used to represent it, or two bits.

Shannon entropy

- $q_i > 0$: the probability of event x_i from a space X of n possible events ($1 \le i \le n$, $\sum_i q_i = 1$). $x_i = P_i$ will be a password chosen from a space of n allowable passwords, with the set of passwords chosen by a system
- A random variable X takes values $x_i = P_i$ with probability q_i according to a probability distribution D_X: D_X models the probability of users choosing specific passwords
- Shannon entropy: $H(X) = H(q_1, \ldots q_n) = \sum_i q_i \lg(1/q_i) = -\sum_i q_i \lg(q_i)$ lg=base-2 logarithm. H(X) measures the average uncertainty of X=minimum number of bits needed on average

Interpretation of entropy

- For each x_i:$I(x_i)=-lg(q_i)$ as the information conveyed by event $\{X_i=x_i\}$
- The less probable an outcome, the more information its observation conveys; observing a rare event conveys more than a common event and observing an event of probability 1 conveys no information
- The average (expected value) of the r.v. I is $H(X)=E_X(I_X)=E_X(-lg(q_i))$
- Entropy properties:
  1. $H(X)\geq 0$. minimum 0 occurs only when there is no uncertainty at all $q_i=1$
  2. $H(X)\leq lg(n)$ all $q_i=1/n$ (uniform distribution = flat)
  3. If $q_1<q_2$, if we increase q_1 and decrease q_2 so that $q_1=q_2$, $H(X)$ increases

Guessing function: which single password has highest probability?

GUESSWORK (GUESSING FUNCTION). With notation as above, let $q_i \geq q_{i+1}$, modeling an optimal guessing-attack order. The *guessing index* $g(X)$ over a finite domain $\mathcal{X}$ assigns a unique index $i \geq 1$, called a *guess number*, to each event $X = x_i$ under this optimal ordering. Then for $X \xleftarrow{R} \mathcal{X}$ (X drawn randomly from the event universe according to distribution D_X above), the average (expected) number of guesses needed to find X by sequentially asking, in optimal order, "Is $X = x_i$?" is given by the *guessing function*

$$G_1(\mathcal{X}) = E[g(X)] = \sum_{i=1}^{n} i \cdot q_i \quad \text{(units = number of guesses)} \tag{3.7}$$

Like $H(X)$, G_1 gives an expectation *averaged over all events* in $\mathcal{X}$. Thus its measure is relevant for an attack executing a *full search* to find all user passwords in a dataset—but not one, e.g., quitting after finding a few easily guessed passwords. If $q_i = 1/n$ for all i,

$$G_1(n \text{ equally probable events}) = \sum_{i=1}^{n} i \cdot 1/n = (1/n)\sum_{i=1}^{n} i = (n+1)/2 \tag{3.8}$$

since $\sum_{i=1}^{n} i = n(n+1)/2$. Thus in the special case that events are equiprobable, success is expected after guessing about halfway through the event space; note this is *not* the case for user-chosen passwords since their distributions are known to be heavily skewed.