Computer Security

Authentication

Access to a computer

- user presents an identifier (name, *login*)
- system demands a confirmation (e.g. password matched to the identifier)

Remote communication

- party1 sends identifier to party2
- party2 challenges party1 with a fresh number that should be enciphered (e.g. with a predefined shared key)

Definition

- binding of an identifier to a subject
  - or: certification of an user's identity
- sometimes : certification of a physical place, e.g. machine (origin of a communication)...

Protection basics

Ultimate purpose

- provide **access control** to resources (e.g. users' information or devices)
  - by building secure channels for their handling:
    - communicating them, storing them
  - secure, with properties
    - main: confidentiality, integrity and authentication
    - secondary: anonymity, forward secrecy, etc.

But, usually, specifically

- *access control* is (just) the enforcement of system's policy on resources' usage[1]
- before being exercised, two phases must be successfully completed:
  - *authentication*
  - *authorization*

1 so, a security policy must be in place: who can do what, how and when!

Definitions

- ***Subject***, *Entity* (or group of entities) - *S*
  - physical person, “active” system's agent[1]
- ***Object*** (or class of objects) - *O*[2]
  - resource that undergoes accesses (or actions) from *Subject*
- ***Information*** (data kept in object) - *I*
  - internal content or state of *Object* to be protected
- ***Action*** (access)
  - utilization of *Object* by *Subject* (e.g. "read" object, "delete" object...)

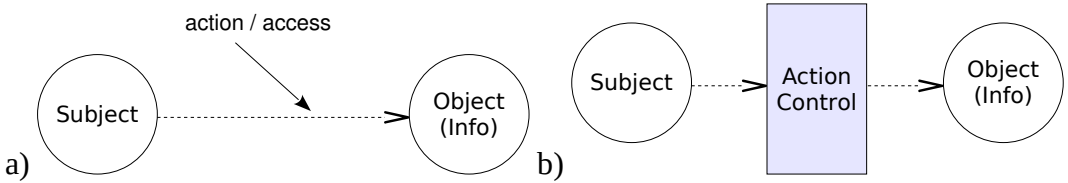


Fig. Acting on an a) unprotected, b) protected object (e.g. to get or change its Information).

1 such as an operating system's *process*
2 Sometimes a *Subject* is an *Object* too, as some actions are performed on *Subjects*.

...Definitions...

- ***Authentication***
  - binding of an identifier to a subject
- ***Authorization***
  - giving permission to a subject perform an action on an object
- ***Access Control***
  - verification that access to object is in accordance to authorization

Sequence of access operations on protected object

- authentication of *Entity*
- retrieval of authorization for accessing the *Object*
- control of the *Subject*'s access to the *Object*
  - the *Object* can be accessed directly or through a resource's server

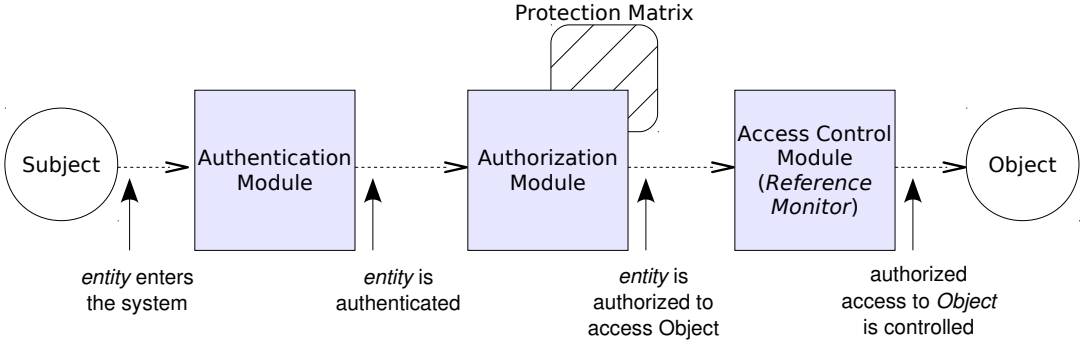


Fig. Recommended procedure for a Subject to access an Object.

Accessing an object: more typical procedure

- theoretical distinction between authorization and access control is not made
  - both name the *process of control of authenticated entities' actions*.

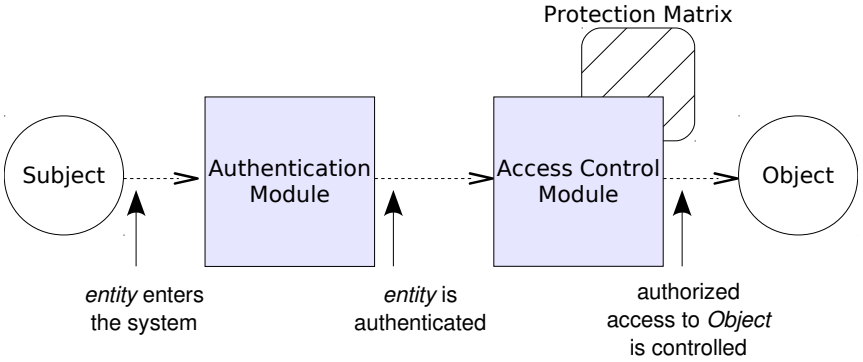


Fig. Typical, simplified, procedure for a Subject to access an Object: authorization and control are run in simultaneity.

Enterprise/Institution governance

- Identity Management, IdM:
  - user accounts, identity life-cycle, synchronization of data in departments
  - «*Who are you*?»
- Identity and Access Management, IAM:
  - authentication, authorization, access control, session management
  - «*Can you login? What can you access?*»
- Identity Governance and Administration, IGA:
  - oversight, policy, compliance (e.g. with GDPR[1]), auditing, risk management
  - «*Should you still have that access?*»

1 European Union's *General Data Protection Regulation*

Authentication

- assuring the identity of the entities involved
  - binding identifiers to subjects
- sometimes: certifying a location
  - e.g. of machine[1] in the Net, machine's or user's geographical position
- authentication **system's operation**: two *phases*
  - setup:[2] generation and storage of subjects' authentication data in system
    - repeated whenever user changes own authentication data
  - usage:[3] normal procedure for authentication of subjects
- authentication **operation**: two *steps*
  - presentation (of subject) [sometimes: *identification*[4]]
  - validation (proof of authenticity) [sometimes: *authentication*]

1 origin of a communication...
2 or registration, or enrollment
3 or authentication (!)
4 Note: this occasional use of "identification" is unfortunate. In reality, identification is the process of binding an identifier to an individual, as yet unknown (i.e. for whom no label, or name, was yet presented).

Authentication system deployment[1]

- **setup phase**: [FIG]
  - generation and storage of subjects' authentication data in system
  - (repeated whenever user changes own authentication data)
- **usage phase**: [FIG]
  - day-to-day procedure for authentication of subjects

1 The following represents the most known procedure and type of data stored; actually, there will be plenty of practical deviations.

Set up phase

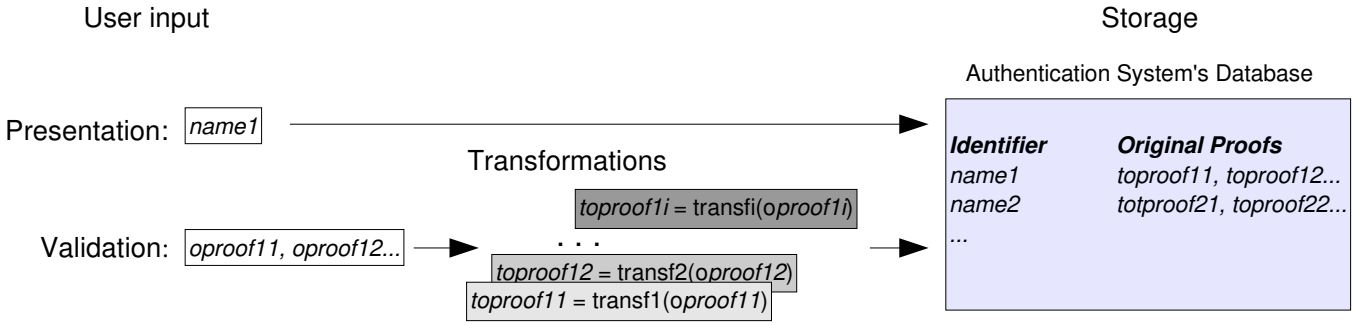


Fig. **Setting up** an authentication system: generation and storage of original proofs.
(Notice that what is usually stored are transformations of original proofs.)

Daily Usage phase

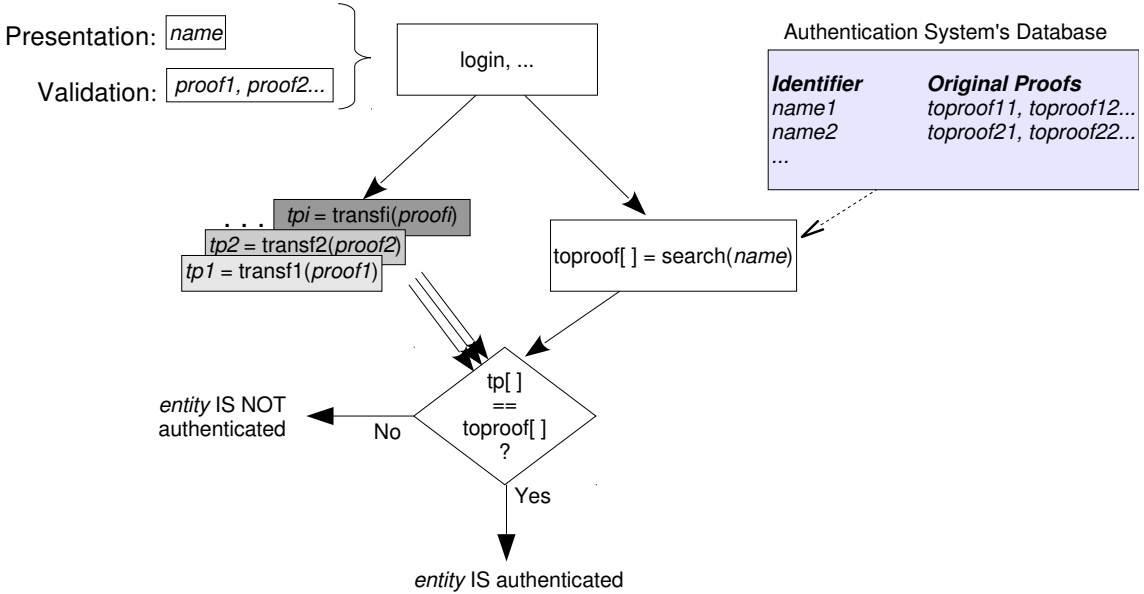


Fig. **Using** an authentication system: validating the proofs, comparing them with those stored in the setup phase.

Validation's methods

- proof by possession:
  - of knowledge: e.g. knowing a personal password
  - of object: e.g. having a personal card
  - of passive property: e.g. having a specific fingerprint
  - of ”active” property (trait): e.g. keying with a certain speed or hitting force
- proof by origin (...): e.g. request comes from a predefined machine or geographical place

More usually

- a different terminology is used for the validation methods: proof by
  - knowledge
  - possession
  - property
  - trait

Proof by (possession of) knowledge validation

Memorizable information

- especially important for *in-person* authentication
- system asks:
  - for two strings: presentation and validation (e.g. `loginname` & `password`)
  - one or more questions whose answers the user should give

Dynamic *challenge-response exchange*

- especially important in remote authentication
- system presents a *never-seen-before* value that the user has to:
  - (**secret algorithm**) - process[1] in a secret way and return the result
  - (**private key**) - process in a public way with the help of a private (secret) key[2] and return the result

1 usually by means of a computing device
2 cryptographic key!

Remote authentication

- user's physical intervention is not possible (or required)
  - presentation by non-physical identifier
  - validation by *proof of knowledge*, typically of challenge-response type [FIG]
- based on the use of pre-distributed keys
- generally, use *nonces*

Nonce:

- piece of data that is both: [1]
  - fresh
  - not guessable (random)
- random number generated when about to be used
- meant for binding two messages in a challenge-response sequence

1 the NOnce, Number Once, designation is misleading, as in most applications, a nonce must also be non-predictable!

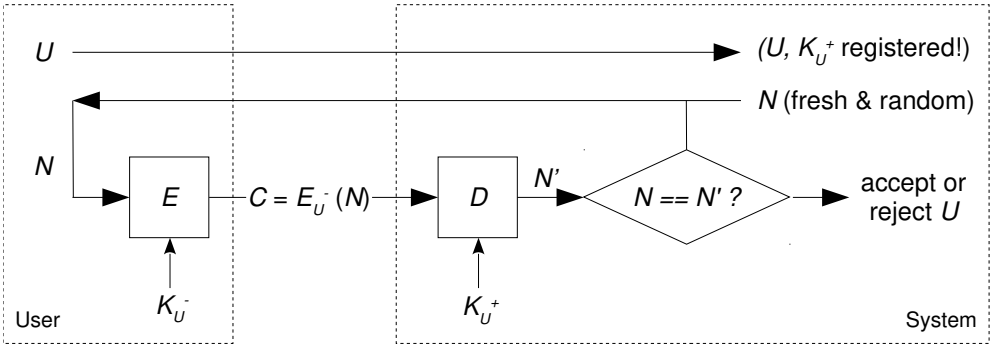


Fig. Remote authentication with asymmetric cryptography.

Challenge-response with smart card

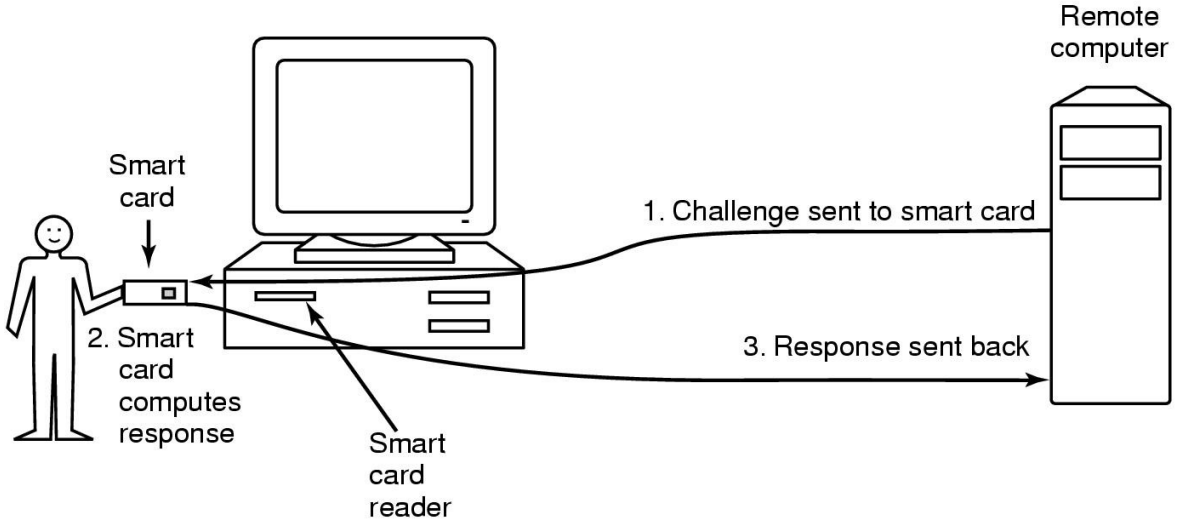


Fig. Example of authentication by challenge-response technique using a *smart card*. (in Tanenbaum, ...)

Authentication Taxonomy

Perspective	*Variant*	*Examples*	*Usability (convenience)*	*Efectiveness (security)*
Validation type	Knowledge	password, PIN, question	good	low
	Possession	smart card, hardware token, phone	high	reasonable
	Inherence	fingerprint, face recognition, iris scan	good	high
	Behavior	typing rhythm, phone motion	low	good
	Location	GPS coordinates, IP address	variable	high
Mechanism	Password	(normal) password	good	low
	One-time password	time-based (TOTP), counter-based (HOTP)	high	high
	Passwordless	magic link, passkey (FIDO2)	high	high
	Cryptography	challenge–response, signed token	reasonable	high
Scope	Local	personal computer	good	good
	Centralized	SSO, directory service (e.g. LDAP)	high	good
	Federated	OpenID Connect, SAML	high	good

Examples

- One-Time Passwords (OTP)
  - Time based OTP Algorithm (TOTP)
- biometrics
- smart cards
  - Personal Identity Verification (PIV)
  - Electronic Identity (eID)
- Secure Remote Password (SRP)
- passkeys
  - Fast Identity Online 2 (FIDO2)
- Single Sign-On (SSO)
  - Federated authentication
  - Kerberos
  - OpenID Connect

One Time Passwords (OTP)

- *Lamport's hash*!
  - Implementation: *OTP System*, IETF STD 61 (orig.: *S/Key System*, RFC1760)

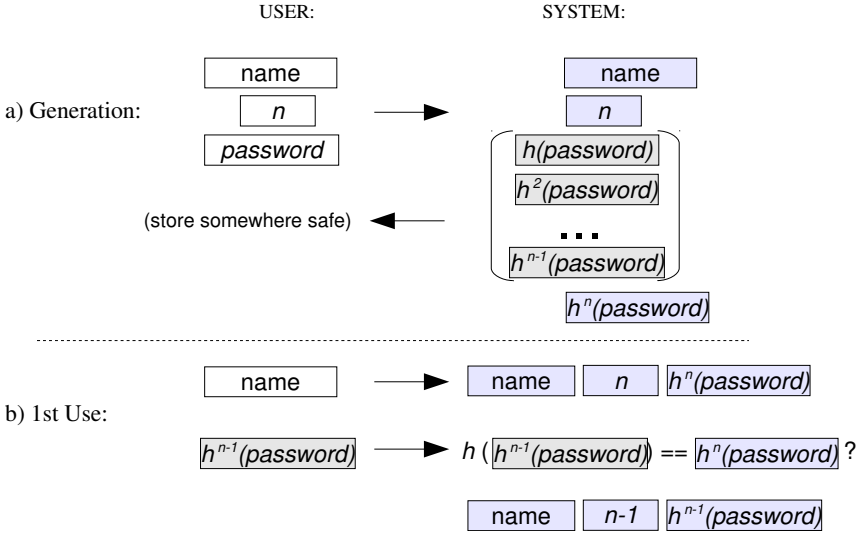


Fig. The Lamport's system of one-time passwords.

TOTP – Time based OTP Algorithm[1]

- TOTP – Time based OTP Algorithm (RFC6238)
  - extension of HOTP – HMAC based OTP Algorithm (RFC4226)
  - used in a two-factor authentication on the Internet
    - one factor can be a simple password
    - the second is the one-time value (password) generated by the algorithm
- basically, relies on
  - secret shared by client and server[2]
  - integer[3], derived from a time reference kept in synchrony by client and server
  - computation of an hash[4], function of previous items, by client and server [FIG]

1 When used as a "2nd factor authentication", it is quite common to refer to TOTP as, simply, *authenticator*!
2 An (authentication) server will have to store a different secret for each (authentication) client.
3 one-time password
4 through HMAC

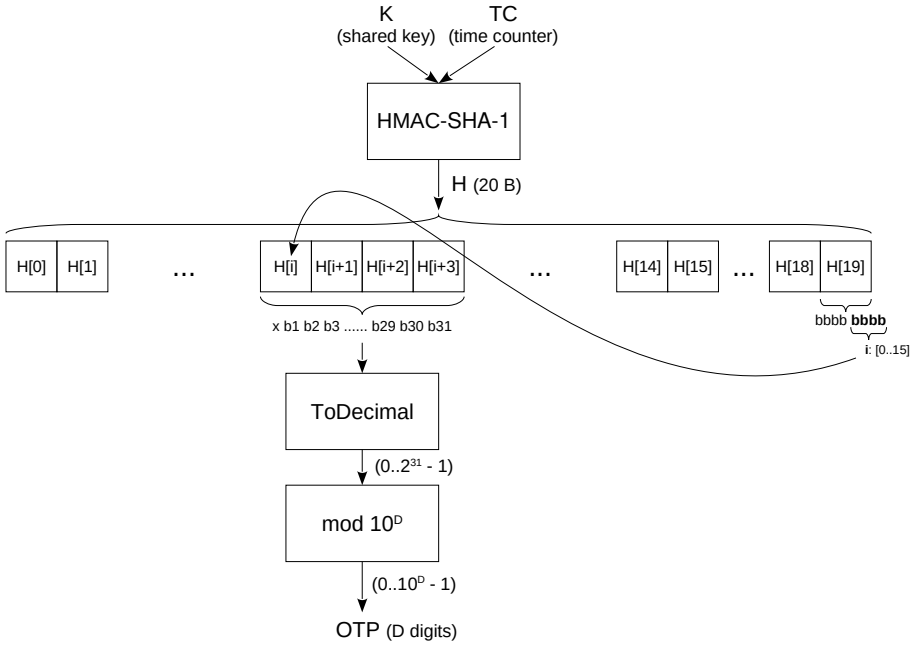


Fig. OTP production in TOTP, RFC 6238.

...TOTP – Time based OTP Algorithm...

- actors
  - user
    - owner of a device that produces a token[1], that wants to be authenticated
  - device
    - hardware or software that keeps secret and time and produces tokens
  - prover
    - i.e. token that is the means of authentication
  - verifier
    - authentication (or validation) server that will perform the authentication

1 In an unfortunate usage, the token is called "prover" in RFC6328. It really is the OTP value that both user and server should independently calculate for each authentication.

...TOTP – Time based OTP Algorithm...

Setup phase (registration)

- typically, server produces (and stores) a random secret and establishes a step value for the calculation of intervals of time, that should allow for some variation of time synchronization
  - quite commonly, server builds a QR-code that will be shown to user
- user will receive parameters (secret and step value) and install them in local authenticator device (e.g. cellphone)[1]

Usage phase (authentication)

- user authenticates (locally) its local OTP device
- local device produces a token (one time password, e.g. 133057)
- user contacts (authentication) server and presents itself
- user gives token to server

1 of course, devive must have installed adequate, TOTP software!

Secure Remote Password (SRP)

- client proves knowledge of password to server without ever sending the password (or even a hash of it) over the network
- server does not store a password-equivalent secret, but rather a "verifier"
- example of Password-Authenticated Key Exchange[1] (PAKE)
- uses Diffie–Hellman-style key exchange

NOTE: *although sometimes said to be a "zero-knowledge system", in fact, it is not because of its mathematical constraints and susceptibility to specific attacks. E.g. a stolen authentication database can be used to impersonate the server!*

Setup phase (registration)

- client user chooses *password* and *salt* and calculates $x = hash\ (salt, password)$
- client sends server [*user, salt,* $v\ (= g^x \bmod n)$, g, n][2]
- authentication server stores [*user, salt,* $v\ (= g^x \bmod n)$]

1 origin: *Encripted Key Exchange: Password-based Protocols Secure Against Dictionary Attacks*, 1992, M. Bellovin, M. Merritt
2 Standard specifies that both client and server have somehow previously agreed on N (large *safe* prime), g (generator modulo N), cryptographic hash function, k (multiplier parameter).

Usage phase (authentication)

- client user sends server: [username, A (= $g^a \bmod n$)
- server replies: [salt, B (= $kv + g^b \bmod n$)][1]
- both compute:
  - $u = hash\ (A, B)$
- client also computes:
  - $K = (B - kv)^{a+ux} \bmod n \dashrightarrow = (g^b)^{a+ux} \bmod n = \boldsymbol{g^{(a+ux)b} \bmod n}$
  - $\boldsymbol{K_{CS}} = hash\ (K) \dashrightarrow$ *session key*
- server also computes:
  - $K = (A \,.\, v^u)^b \bmod n \dashrightarrow = (g^a \,.\, g^{xu})^b \bmod n = \boldsymbol{g^{(a+ux)b} \bmod n}$
  - $\boldsymbol{K_{CS}} = hash\ (K) \dashrightarrow$ *session key*
- one sends K_{CS} (*nonce*), other replies *nonce*![2]

1 Both *a* and *b* are ephemeral and each is secret to client and server, respectively.
2 This last step was originally different, but conceptually identical: it proves that both parties know the session key!

Biometric authentication

- authentication data can be:
  - fingerprints
  - eyes (iris or retina)
  - palm (lines or veins)
  - voice
  - facial features
- problems:
  - false positives and false negatives
  - biometric data cannot be changed if stolen
- focus:
  - converting raw physical data into a protected, mathematical format
    - efficient for matching
    - difficult to invert (regenerate physical data)

...Biometric authentication...

Setup phase (registration)

- special sensor collects raw data from user
  - e.g. ridges (fingerprint scanner), face (camera), voice sample (microphone)
- preprocessing removes noise and normalizes input (e.g align orientation)
  - for consistency across captures
- feature extraction produces "feature vector" (mathematical representation)
  - e.g. ridge endings, bifurcations (fingerprint), distance between eyes, nose (face), texture patterns (iris)
- template creation encodes features and provides non-reversibility
  - e.g. transformation functions, hashing
- resulting data is stored, with special care as biometric data is non-revocable[1]
    - e.g. use of cryptography, *secure enclaves*[2], sharding[3]

1 except for "cancelable biometrics", where some distortion transformations controlled by secret key are applied; if compromised, transform or key are changed
2 special harware isolated from the main processor that keeps data secure even if system code becomes compromised
3 splitting of data in pieces to be distributed across multiple servers

...Biometric authentication...

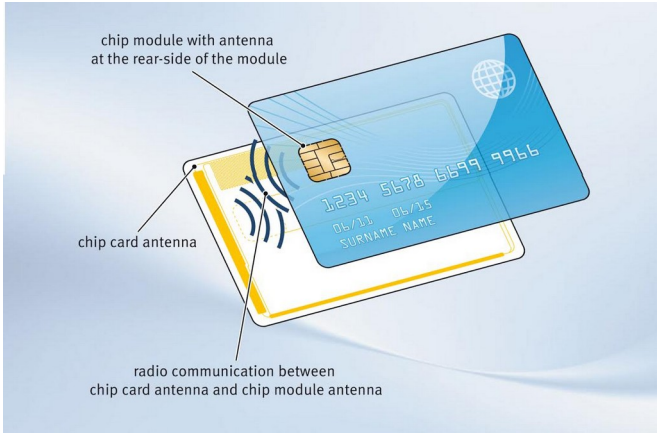
Usage phase (authentication)

- special sensor collects raw data from user[1]
- data is processed in exact same way as in setup phase
- resulting data is matched with authenticating database
  - but... matching is probabilistic, not exact, so
    - output is given as similarity score (%)
    - acceptance (authentication) if above threshold[2]

1 Interesting spoofing case: German minister fingered as hacker 'steals' her thumbprint from a PHOTO (2014).
2 that usually has to be adjusted to different environments and users

Smart Cards

- Integrated Circuit with memory, processor, and I/O capabilities
  - hardware needed!
    - card reader!
  - actions
    - perform PIN[1] authentication
    - generate one-time passwords
    - support biometric authentication
    - respond to cryptographic challenges
  - types
    - contact
    - contactless (NFC/RFID)[2]
    - dual-interface ;-)



1 Personal Identification Number
2 Near Field Communication / Radio-Frequency Identification. Generically, they are similar, using radio waves for data transfer. In practice, they use different specifications such that NFC has shorter range, communicates two-way, supports high security applications.

...Smart Cards...

Benefits	***Challenges***
Phishing resistance (remote attackers cannot steal credentials)	High upfront cost (requires cards, readers and PKI)
Offline capability (can authenticate without Internet connection)	Administrative overhead: (managing lost cards and expiring certificates)
Consolidated access (one card for building entry and PC login)	Hardware dependency (needs a reader at every workstation)

Technical Operation:

- Power-Up
  - reader powers card (by pin contacts or RF)
  - card boots OS, sends "Answer To Reset", communicating state and capabilities
- Communication (Application Protocol Data Units)
  - short transactions: reader requests - card answers
  - stateless or minimally stateful interactions
  - reader command examples:
    - SELECT (file/app), VERIFY (PIN/password),
    - GET DATA, GET CHALLENGE (nonce),
    - GENERAL AUTHENTICATE (cryptographic ops)
  - card responses examples:
    - data
    - status codes (success, failure with reason)

Personal Identity Verification (PIV)

- strong identity credentials for people accessing federal systems and facilities
  - NIST's FIPS 201 standard
- basically, relies on
  - strict issuance verification process (identity proofs, background checking)
  - smart card (cryptographic keys, digitized fingerprint and photograph, PIN[1]...)
  - multiple factor authentication:
    - card possession, PIN knowledge, cryptographic challenge-response, biometrics
- actors
  - issuance entity (corporate, government) & Identity Management System (IDMS)
  - user, card holder
  - smart card (includes active circuits)
  - Public Key Infrastructure (PKI)
  - Identity Provider (IdP), that could be the IDMS

1 Personal Identification Number

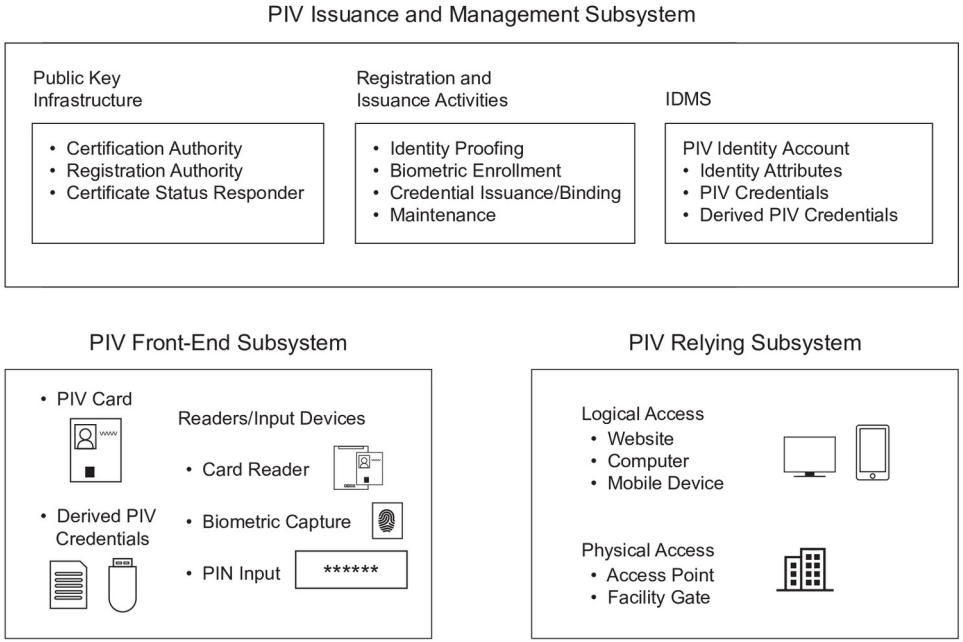


Fig. PIV System Overview (in FIPS PUB 201-3).

...Personal Identity Verification (PIV)...

Setup phase (registration)

- user is subjected to strict identity and integrity verification process
  - identity proofs, background checking...
- card is issued with
  - public cryptography credentials [1]
  - digitized fingerprint and photograph
  - Personal Identification Number (PIN)
- enterprise IDMS (supposing is also the Identity Provider), usually, stores:
  - user general information (e.g. Cardholder Unique Identifier)
  - digital certificate
  - credential status

1 Key pair should be generated on-card, by its secure chip. Public key, via Certificate Signing Request, is sent to Certificate Authority and resulting Digital Certificate copied to the card.
Note that key pair could be generated externally, in Hardware Security Modules (HSM) or even in centralized key management systems!... :-(

Usage phase (typical multi-factor authentication)

- user inserts card in card reader, and answers PIN question
- user provides biometric data through adequate sensor
- card answers cryptographic challenge of authentication server
- if every authentication factor is successfully overcome, user is authenticated

Note:

- for utmost security (including privacy), all of the authentication challenges should be answered with data kept and computation done in the card.
- But, of course, there could be exceptions due to special enterprise/institution's policy...

Electronic Identity (eID) systems

- general term for any digital identity system
  - allows individuals, organizations or devices to prove their identity electronically
- often refers to national schemes (like those under the EU's eIDAS regulation)[1]
  - give citizens digital equivalent of physical ID cards but much more versatile
  - allow access to public services, sign legally binding documents, and prove their identity online
  - e.g. Portugal citizen card (& mobile key) system

	National eID	***PIV (FIPS 201)***
Governance	national/regional	U.S. federal standards
Primary Users	citizens of country	U.S. federal employees, contractors
Use Cases	Filing taxes, digital signatures, passports, drivers license	Physical building access, network login, document encryption
Trust Model	usually, government-backed	PKI managed by federal agencies
Hardware	smart card, mobile device	smart card or, more rare, mobile device
Issuance	civil registries, post offices	rigorous, in-person identity vetting

1 EU: European Union; eIDAS: Electronic IDentification, Authentication and trust Services

Passkeys

- modern[1] authentication replacement for passwords
- uses public-key Cryptography
  - near user: device (or "authenticator")[2] with private key
  - far, in website/server: public key
  - authentication with "normal" challenge-response remote protocol
- so, usually is used for website/appserver login
  - distinguishing feature: public-key credentials are different for each website!
- currently, came alive with FIDO2[3] standard for website authentication
  - essence of "passkeys" paradigm will be presented here
  - FIDO2 will be presented in a following section

1 at last!...
2 *Authenticator* is usually a device (hardware + software) that handles user's authentication credentials and performs *immediate* (near user) authentication. But it can be just software, internal to the browser or the operating system.
3 FIDO: Fast Identity Online; the 2 means the second major generation of the FIDO authentication architecture framework.

...Passkeys...

Setup phase (registration)

- user, through web browser, accesses compliant website/webserver[1] for the 1st time
- being 1st time access, website login interface asks for creation of *passkey*
- browser passes request to authenticator
- authenticator creates new public-key pair, stores private-key and passes signed public-key to browser[2]
- browser forwards attestation (signed user public-key) to website/webserver
- if verification of attestation succeds, website/webserver stores the public key associated with client/user for future authentications
  ---> see [FIG]

1 Usually, the website/webserver is named "relying party" as it relies on an external party (identity provider) for the authentication of user.
2 If available, the public key is signed by a special "attestation key" burned into the hardware by its manufacturer and is linked to a certificate that confirms the device's make and model; otherwise, public key is signed with pair's private key.

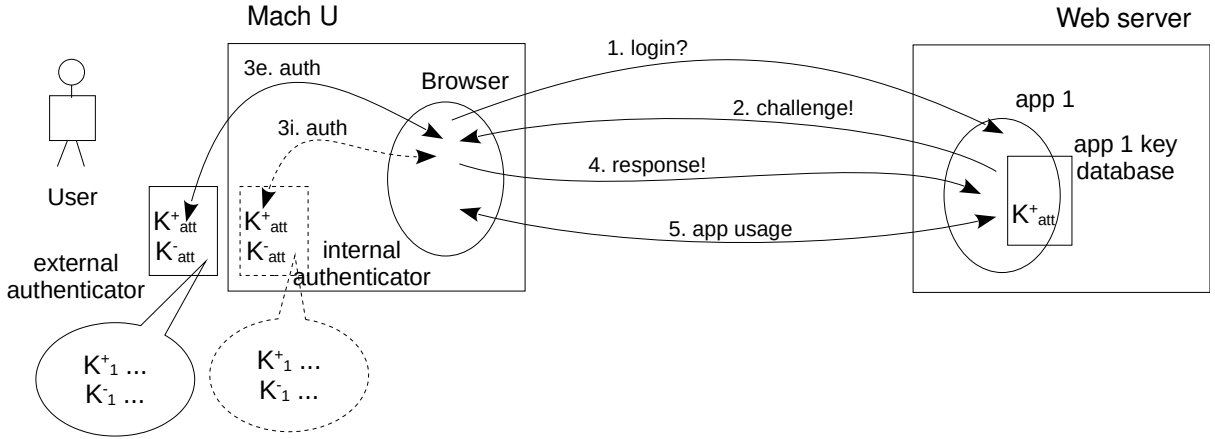


Fig. General operation of passkeys (adaptable to both registration and usage).
The authentication can be performed with external or with internal authenticator.

...Passkeys...

Usage phase (authentication) [FIG]

- user, through web browser, accesses compliant website/webserver
- website login interface starts passkey authentication by sending challenge (nonce)[1]
- browser passes challenge to authenticator (internal or external)
- authenticator[2] grabs user's private-key specific of website and signs nonce
- browser completes authentication protocol by responding to website with signed nonce
- (if success) user is logged in website/webserver

1 there could be several authentication factors...
2 authenticator could demand from user a local authentication (e.g. biometrics/PIN)

FIDO2

- secure, phishing-resistant authentication procedure
  - based on public-key cryptography
  - enabling passwordless logins
  - using local authenticators (biometrics or security keys)
- core operations:
  - registration to create credentials
  - authentication to verify identity, based on unique key pairs for every service
- includes protocols
  - WebAuthn – W3C's Web Authentication API
  - CTAP – Client To Authenticator Protocol[1]

1 used with authenticators external to operating system or to browser

Entities

- User (human)
  - of browser, tries to sign in compliant website/webapp
  - provides "local" authentication (biometric / PIN / touch...)
- Client (user agent, browser)
  - software residing on the user's computer or mobile device
  - handles all interaction with website/webapp and "authenticator"
- Relying Party (web service or mobile app)
  - service provider (e.g. Google, GitHub, banking site) that wants to authenticate user before providing its service
  - for authentication relies on an external system, previously accepted[1]
- Authenticator
  - logical or physical device that creates and stores the user's private keys and performs cryptographic signatures

1 right since installation or from re-configuration of service

...FIDO2...

Setup phase (registration)

- user accesses through browser a FIDO2-enabled website/webapp[1]
  - and tries login interface (Web)
- as Relying Party, website delegates authentication[2] to browser or Identity Provider
  - if theres is an Identity Provider, app sends redirection request to browser (Web)
- Identity Provider searches its authentication database for user; if 1st access, sends registration request to browser with (WebAuthn):
  - Relying Party identifier (e.g. website's domain)
  - user identifier (e.g. email address)
  - nonce
- browser contacts authenticator (e.g. USB key, mobile phone) and mediates its interaction with user (CTAP)
- ... *continues...*

1 also known as *Relying Party*
2 there could be several authentication factors, some of them performed by the Relying Party (webapp) itself...

...FIDO2...

- authenticator:
  - verifies user with proximity methods (e.g. PIN or biometrics)
  - generates a new asymmetric key pair, unique for the Relying Party ID
  - internally stores:
    - the new private key
    - the user identifier (e.g. email address)
    - the hash[1] of the Relying Party identifier (e.g. website's domain)
    - user credential identifier[2]
  - signs the nonce and the public-key[3]
  - returns both signatures to browser
- ... *continues...*

1 for privacy...
2 opaque value, built with user info (e.g. email address), for easy reference to user ant its public key in subsequent authentications.
3 If available, the public key is signed by a special "attestation key" burned into the hardware by its manufacturer and is linked to a certificate that confirms the device's make and model; otherwise, public key is signed with pair private key.

...FIDO2...

- browser forwards signed nonce and user's attestation (signed public-key) to Identity Provider (WebAuthn)
- if verification of attestation succeeds, Identity Provider stores, for future authentications:
  - the user credential identifier
  - the public key associated with user
- and responds to Relying Party[1], by means of another redirect request mediated by browser (Web)
- if website/webapp is satisfied with authentication, user is logged in for that session (Web)

1 possibly, with an *identity token*

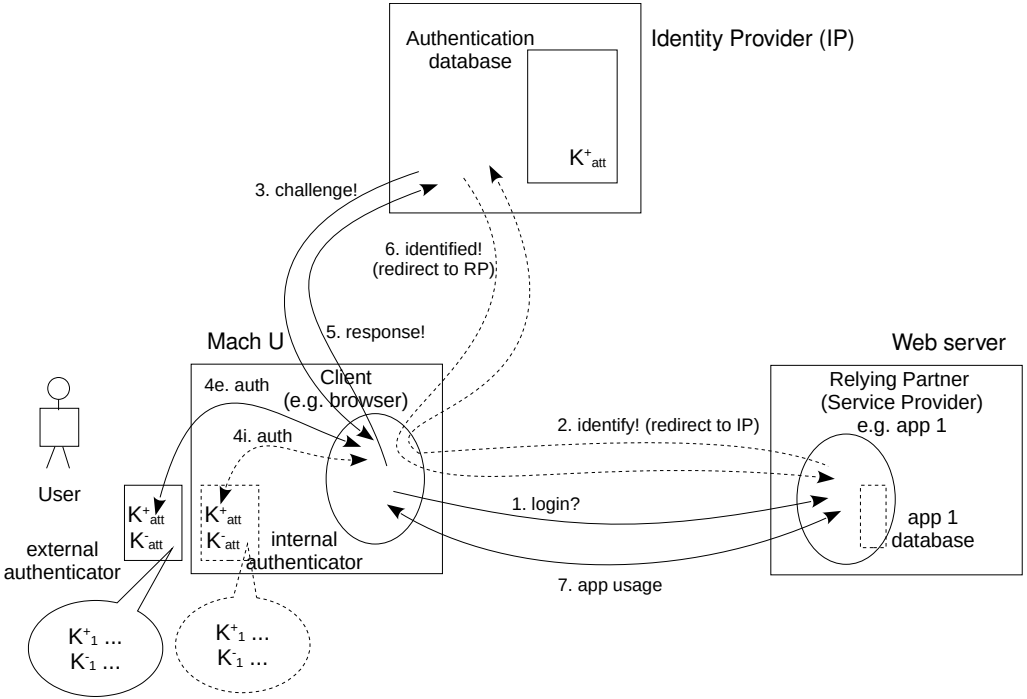


Fig. FIDO2 authentication between a user and a Relying Party that trusts an Identity Provider.

...FIDO2...

Usage phase (authentication)

- user accesses through browser a FIDO2-enabled website/webapp[1]
  - and tries login interface (Web)
- as Relying Party, website delegates authentication[2] to browser or Identity Provider
  - if theres is an Identity Provider, app sends redirection request to browser (Web)
- Identity Provider searches its authentication database for user and retrieves data
  - sends authentication request with nonce to browser (WebAuthn)
- browser contacts authenticator (e.g. USB key, mobile phone) and mediates its interaction with user (CTAP)
- authenticator:
  - verifies user with proximity methods (e.g. PIN or biometrics)
  - grabs user's private key, signs the nonce with it and returns the result to browser
- ... *continues...*

1 also known as *Relying Party*
2 there could be several authentication factors, some of them performed by the Relying Party (webapp) itself...

...FIDO2...

- browser forwards signed nonce (assertion) to Identity Provider (WebAuthn)
- if verification of signature succeeds, Identity Provider responds to Relying Party with it an *identity token*, by means of another redirect request mediated by browser (Web)
- if website/webapp is satisfied with authentication, user is logged in for that session (Web)

Attestation vs. Assertion

	Attestation	*Assertion*
when	during registration	during login
purpose	proves device type and origin	proves user's identity using passkey
how	signature with attestation's private key	signature with user/website's private key

Single Sign-On,* SSO

- single, initial authentication for all sessions on all machines
- allows use of cryptographic keys
- possible implementations:
  - private solution, e.g. password wallet
  - enterprise service (e.g. Kerberos or LDAP)
  - federated authentication
- problems:
  - safe keeping of single (or master) password (even with wallets!)
  - session hijacking
    - partial solution: periodical renewal of authentication
  - logout difficulty
    - it might not be easy to guarantee logout in every system

* PT: autenticação única

Federated authentication

- allow users to access
  - multiple applications in
  - multiple systems (even institutions) with
  - single set of login credentials
- authentication flow general structure[1]: [FIG]
  - User Agent (e.g. browser) interfaces with human and Service Providers (SP)
  - Service Providers trust authentication to Identity Providers (IdP)
    - from them receiving
      - tokens / assertions, proving identity information[2]
- common technologies:
  - SAML (Security Assertion Markup Language)
  - OpenID Connect (OIDC) (with OAuth 2.0...)
  - ---> see comparison table ahead

1 of course, depends on protocol
2 sometimes, also authorization...

...Federated authentication...

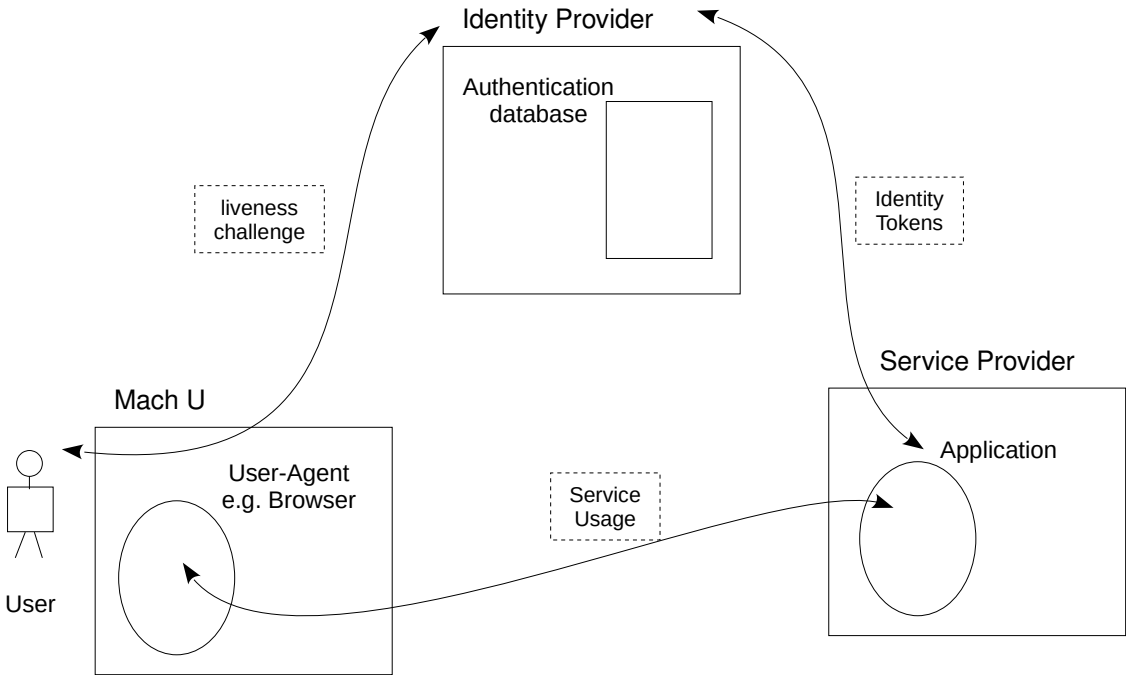


Fig. Federated authentication: general perspective.

...Federated authentication...

Setup phase (registration)

- global
  - Identity Providers and Service Providers establish trust relationship and agree on:
    - attribute mappings (e.g., email - username)
    - cryptographic data (e.g. algorithms, public keys)
    - endpoints (e.g. URLs for login, tokens, logout...)
- user specific
  - special registration interface and operations
    - details depend on specific system (see OpenId Connect example, ahead)
    - result is setting of user profile in Identity Provider's database

...Federated authentication...

Usage phase (authentication)

- user tries to log in to an app or a website
- the app redirects user to an Identity Provider (e.g. Google, Microsoft...)
- user logs in there (if not already logged in)
- Identity Provider performs authentication and sends a *secure token* back to the app
  - if necessary, Identity Provider interacts with User (e.g. for liveness proof)
- app verifies integrity of token and grants access to user

SAML vs OpenID Connect

Aspect	*SAML*	*OpenID Connect*
Era	older enterprise standard, stable usage but aging	modern web/cloud standard, dominant growth
Base framework	XML	OAuth 2.0 + JSON/JWT
Data format	heavyweight XML assertions	lightweight JSON/JWT ID Tokens
Typical environment	enterprises, institutions	web/mobile/cloud environments
Browser-centric	yes	less browser-dependent
Mobile API integration	weak, awkward, high complexity	strong, native, low complexity

Pointers...

- "**Federal Information Processing Standard 201 (FIPS 201)**", 2005-22, National Institute of Standards and Technology (NIST)
  - csrc.nist.gov/pubs/fips/201-3/final
- "**The Stanford SRP Homepage**", Tom Wu, 200?, Stanford University
  - srp.stanford.edu/index.html
- "**FIDO 2.0: Overview**", FIDO Alliance, Review Draft, 04 October 2017
  - fidoalliance.org/specs/fido-v2.0-rd-20170927/fido-overview-v2.0-rd-20170927.html