c = m^e \mod n

Authentication Tokens
(A Survey)
The three classical authentication factors

1. **Something you know**
   - E.g., password, password-recovery question

2. **Something you are**
   - Biometric

3. **Something you have**
   - Authentication token

- **But there are also…**
  - Someone you know
    - "Social authentication"
    - E.g., Facebook Trusted Contacts
  - Somewhere you are
    - Location as an authentication factor
  - Etc., etc.
What can we do about the password problem?
Password cracking

(1) Steal $H(P)$

(2) Crack $H(P)$; get $P$

(3) Impersonate user

"Alice"
Eavesdropping

Alice

P

(P (secret)

Eve

h(P)
Sticky notes

Google Glass

Visual capture

Eve

P (secret)

Alice
Malware

E.g., keystroke logger

Alice

Eve

$P$ (secret)
Phishing

At the last reviewing at your Amazon account we discovered that your information is most frauds are possible because we don't have enough information about our clients. Reenter your personal information.

Please follow this link to update your personal information:


(To complete the verification process you must fill in all the required fields)
Social engineering

“Hi, Eve. This is Cornell IT. (Go Big Red!!) A hacker has broken into your account, and we need to change your password...”
Idea 1: User-driven password changes

- Common interval: 90 days
- May help sometimes, but...
  - 90 days is a long time!
- Helps users forget passwords
  - Estimated $150 cost per user per year
    - META group estimate: 1.75 help desk calls a month;
      Gartner group: 30% of calls are for password resets;
      Forester research: $25 / call
  - Makes social engineering worse
Idea 1: User-driven password changes

• How do users change their passwords?
  
  Password1
  Password2
  Password3
  Pa$sword1

Idea 2: One-time passcode token

Alice

- 789128
- 001025
- 330236
- 919511
- 668336

...
A scratch-off variant

- **Pros:**
  - Fits in wallet
  - Recyclable
  - You feel as though you have a chance of winning the lottery

- **Cons:**
  - Winning the lottery just means you can log into your bank account
  - Messy, inconvenient
  - Limited-use
One-time passcode tokens

“Something you have” authentication factor

Many types

(Proof that security can be stylish)
How a time-based token works

$P_T = F(K, T)$

secret key $K$

Alice

$P_T (e.g., 790062)$

$P_T = F(K, T)$

$K$

$T$
Similar for counter-based token

secret key $K$

Alice

$P_c = F(K, C)$ (e.g., 878883)

$P_c = F(K, C)$

$C \leftarrow C+1$

$C \leftarrow C+1$
Adversarial model and security goal?

• Adversarial model:
  • One-time passcode tokens assume an eavesdropping / passive adversary

• Security goal:
  • Assume that the adversary learns an arbitrarily (polynomially) long sequence of passcodes $P_1, P_2, \ldots P_n$.
  • We want adversary not to be able to guess $P_{n+1}$.
  • What does this mean?
    • Ideally, adversary can do no better than random guess at $P_{n+1}$.
    • Looks like what crypto primitive?
What’s the function $F$?

Some (simplified) variants used in practice:

- $F(K, C) = H(C \mathbin\Vert K)$
  - E.g., Old RSA SecurID used proprietary hash (Alleged SecurID Hash Function (ASHF))
  - 64-bit key of 10% of tokens recoverable with two months of passcodes
  - Why did they roll their own?
    - Invented in 1985…

- $F(K, C) = AES_K(C)$
  - E.g., RSA SecurID today (simplified)

- $F(K, C) = HMAC(K, T)$
  - In OATH standard, RFC 6238 Time-Based One-Time Password Algorithm (TOTP)
  - E.g., Google authenticator
What’s the function $F$?

Output needs to be truncated for passcode display

- E.g., $P_C = F(K, C) = H(C || K) \mod 1,000,000$ (for 6 digits)

- OATH approach

- Let $P'_C$ denote *untruncated* passcode

- We'll come back to this…
What happens if Alice pushes the button but doesn’t authenticate?

\[ P_c = F(K, C) \]

\[ P_{c-1} = F(K, C-1) \]
The fix: accept a window of $W$ passcodes
The fix: accept a *window of* $W$ passcodes

Drawback?

- Now adversary can guess any of $W$ passcodes to impersonate Alice
- I.e., window size $W$ gives increases adversary’s success probability by factor of $W$!
- And you’ll still get desynchronized if your six-year-old daughter discovers how fun it is to press the button…
How about challenge-response?

- Desynchronization problems gone!
- Royal pain to use!

\[ P_c = F(K, C) \]
PIN entry
PIN entry

• User also typically enters a PIN
  • Token: “something-you-have”
  • PIN: “something-you-know”
  • Together, they are “two-factor” authentication

• But how do you protect the PIN?
The PIN transmission problem

- A PIN is just like a password
- So Eve can steal it as she stole passwords
- We’re struck with our original problem!
Consider a 6-digit token with keyed-in 4-digit PIN

Is there some way to build a token that displays a 6-digit one-time passcode $Q_C$ such that:
1. $Q_C$ can be checked by the server;
2. PIN can be verified by the server; and
3. PIN isn’t revealed to eavesdropper Eve?

Remember without PIN:
- $P'_C = H(C || K)$ (untruncated)
- $P_C = H(C || K)$ (truncated)
- You should use hash function for simplicity (assume random oracle model)
Good idea that doesn’t work

• How about \( Q_c = H(P_c || PIN) \) (truncated to 6 digits)?

• The hashed value \( P_c || PIN \) is 6 + 4 = 10 digits long
  • 10,000,000,000 possible values for \( P_c || PIN \)
  • Small search space—equivalent to \(~33\)-bit key
    • E.g., Bitcoin mining rig typically achieves over 4,000,000 hashes per second
    • So \( Q_c \) can be cracked and PIN extracted in less than an hour!
What’s done in practice?

- **Idea 1:** Check the PIN on the token.
  - If it’s wrong, don’t emit passcode or else emit incorrect passcode.
- **Idea 2:** \( QC = H(K || C || PIN) \) (truncated) (or \( QC = H(P'c || PIN) \))
  - Why?
- **Idea 3:** \( QC = Pc + PIN \) (digitwise addition)
  - This turns out to be a form of encryption of the PIN under the passcode…
  - E.g.:

\[
\begin{array}{c}
878883 \\
+ 1234 \\
\hline
879017
\end{array}
\]

\( Pc \)  
\( PIN \)  
\( QC \)
Duress PINs

• If user is physically threatened…
• Can enter a second, special “duress” PIN
• E.g., 1234 → 4321
• Server still authenticates user.
  • But it sounds silent alarm, calls police, calls in U.S. Marines, etc.
• Rumored use in ATMs
• Nice idea, but not actually in common use.
Problems with authentication tokens
Man-in-the-middle attacks

- Phishing, malware, social engineering can all capture at least one passcode
- So Eve can impersonate Alice at least once
Lunchtime attacks

- You leave your token on your desk during lunch.
- Mallory steals into your office, breaks open your token and extracts secret.
- Mallory replaces token so you don’t know about attack.
- Mallory uses your passcodes and impersonates you…
Lunchtime attacks

Mallory

• Countermeasures:
  • Tamper resistance
    • E.g., delete key if tampering detected (standard method)
  • Tamper detection
  • Covert channel in passcodes to alert server
Power-analysis attacks

• Counter-countermeasure to tamper protections
• Idea:
  • Collect power traces with minimal device tampering
  • Power traces reflect key-dependent operations
• Differential power analysis (DPA)
  • Idea: Statistically analyze power traces over many computations and a range of inputs
And the counter-counter-countermeasure?

- Counter-counter-countermeasure for DPA attacks
- Idea: Synchronous token-server key update
- Why does this help?
Poor usability

• Things people don’t like:
  • Wearing authentication tokens as necklaces, carrying them everywhere, etc.
  • Transcribing passcodes + PINs
  • Users dislike using tokens for authentication…
Replacing clay pigeons for a game of go-fish

Testing friends’ psychic abilities

Using for game of go-fish
Lost, forgotten, and broken tokens

- The Achilles heel of "something you have"
- How to recover?
  - Help desk call for temporary password
  - Life questions…
Cost

- Tokens can cost $50-60 apiece
- Some lower-cost options available...
  - E.g., Deepnet GridID
Server compromise

- Note: We can't even protect server-side secrets via hashing!
Countermeasure: Distributed Authentication

- Idea:
  - Key $K$ split across two servers
  - Passcode split across two servers
  - Servers verify correctness of passcode through distributed cryptographic operation
  - Compromise of one server does not reveal $K$ or passcodes
  - Extensible to $k$-out-of-$n$

- E.g., Dyadic Security offers as commercial product
- Note: public-key crypto isn't practical because of short passcode lengths!
Other problems

- Mobile devices are vulnerable to malware
- SMS sometimes used; can be compromised in other ways
- Consumers often don’t activate when it’s optional
The future of authentication tokens
The authentication situation is desperate. But Motorola has an answer (two, actually).

Good for teenagers: “... you can be sure that they'll be far more interested in wearing an electronic tattoo, if only to piss off their parents...”

“The pill features a small chip with one switch that uses your stomach acids to activate an 18-bit ECG-like signal inside your body.”

Already FDA approved.
FIDO U2F tokens

• FIDO (Fast IDentity Online) Alliance
• Heavily supported by Google
• Universal Second Factor (U2F)
Web authentication with U2F

Registration

• Independent credentials for each origin \( S \)
• "Origin" is (protocol, hostname, port)-triple
• To register with \( S \), Device generates origin-specific key pair \((SK_S, PK_S)\) and random key handle \( H_S \)
• Sends \((PK_S, H_S)\) to \( S \)
• Stores \( SK_S \) locally with \( h(S) \)
Web authentication with U2F

**Authentication**

- Origin S sends $H_S$ and challenge $c$ to browser
- Browser sends $H_S$, hash $h(S)$, and random challenge $c$ to Device*
- Device checks that $H_S$ is valid and was issued for $h(S)$
- Device digitally signs $c$
- Signature $\text{sig}$ is sent to origin, which verifies it against $PK_S$

\[ \text{sig} = \sum_{SK_s}[c] \]
What problems in SecurID does U2F address?

- **Usability** (partly): Eliminates transcription (partly)
  - Device may communicate via USB, Bluetooth LE, NFC, etc.

- **Server-compromise**: Using public-key crypto

- **Man-in-the-middle attacks**: Origin checking
Hardware protection

- Tamper-resistant secure element (SE) may be used to protect SK$_S$
  - For cost effectiveness, SE has very little memory
  - SK$_S$ is "wrapped" (encrypted) under symmetric key stored in SE
- SE also protects private attestation key that proves device is genuine
What problems does U2F not address?

- *Lost, forgotten, and broken tokens*
- **Cost:** Who's going to pay for these things
  - $18-$79 (for limited edition colors)
- **Man-in-the-browser attacks**
  - E.g., what if Trojan presents incorrect bank transaction details?
- **Usability:** User still needs to insert token, unlock token, press button, etc., etc.
Real usability would require a futuristic device…
Real usability would require a futuristic device....
Like something invented in the 1980s…
It’s now 2016. Why can’t I unlock my computer like my car?
It’s now 2016.
Why can’t I unlock my computer like my car?
Several wireless-authenticator research proposals

• Locking workstation when user walks away

• Proximity-based file-system decryption

• Uber-device for web passwords, screen saver passwords, file system, PINs for consumer devices, etc.
...and several complications

• How to achieve legacy compatibility?
  • E.g., retooling of websites?
• How to balance automation with user confirmation?
  • Bluetooth, 802.11, etc. historically unreliable, especially device-to-laptop connection
• But this is changing...
Nymi

- User authenticates to wristband when putting it on
  - Biometric authentication via heartbeat (EKG)
- Band remains attached -> user remains authenticated
- Motion sensors + NFC, Bluetooth, etc. offer futuristic authentication options
  - E.g., gesture to open car trunk
- "Prosthetic biometric"
"Prosthetic biometrics"?
Last year...

MasterCard and Nymi say they’ve completed the first heartbeat-authenticated mobile payment in the wild.
Will smartwatches be the killer authenticator?

- They already have
  - Physiological sensors
  - Motion sensors
  - Detachment detection
  - NFC + Bluetooth
  - etc., etc.
A killer app for the Apple Watch: Gun control
Interesting related technology: Eddystone-EID

- Eddystone: Google BLE beacon format standard
- EID: Ephemeral ID
- Rotating ID just like that in user authentication token
  - $\text{PRF}_K(T)$, where $T$ is current "epoch" (length at least 512 secs.)
- Addresses both privacy and authentication
  - E.g., Samsonite Track&Go
    - Tracks luggage from mobile app within 70 meters
    - Crowdsourced location of missing luggage
Many interesting research questions

- Formal analysis of U2F protocol
  - (Attestation key looks problematic)
- Physical attacks on Nymi, smartwatches?
  - Detachment
  - Biometric impersonation? Photoplesythmography?
- New ways to detect man-in-the-browser attacks?
  - E.g., can smartwatch video laptop or mobile screen?
- Homework: Can device secretly alert servers when tampered with?