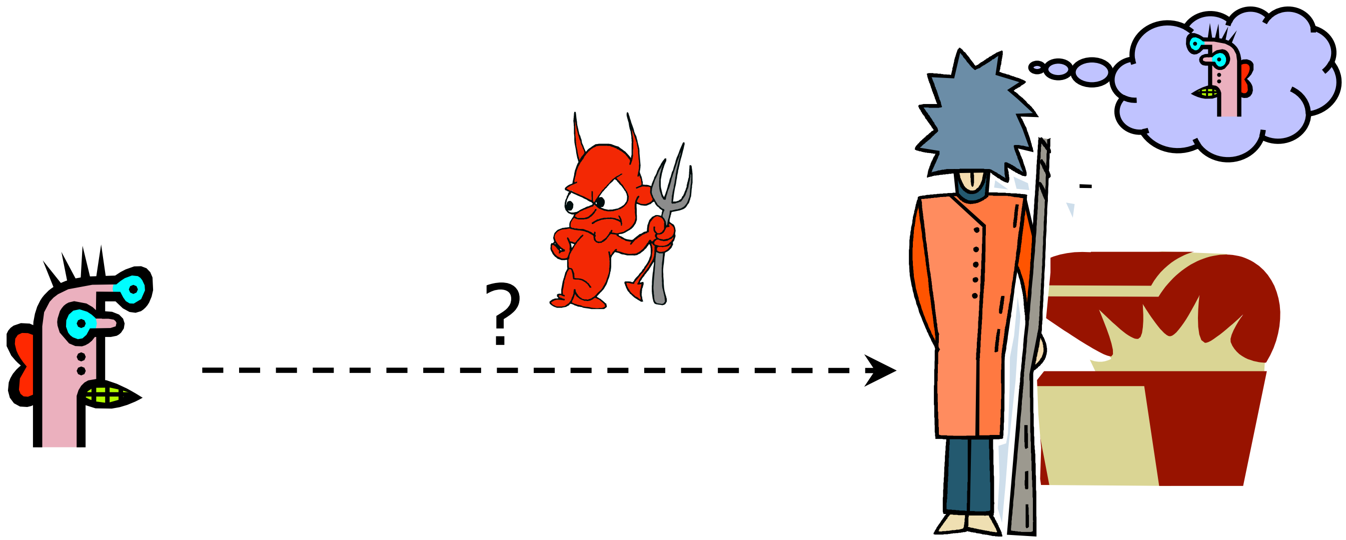
CSE 490K Lecture 8

User Authentication

Tadayoshi Kohno

Some slides derived from Vitaly Shmatikov's

Basic Problem



How do you prove to someone that you are who you claim to be?

Any system with access control must solve this problem

Many Ways to Prove Who You Are

- What you know
  - Passwords
  - Secret key
- Where you are
  - IP address
  - Physical location
- What you are
  - Biometrics
- What you have
  - Secure tokens
- All have advantages and disadvantages

Why Authenticate?

- To prevent an attacker from breaking into <u>our</u> account
  - Co-worker, family member, ...
- To prevent an attacker from breaking into <u>any</u> account on our system
  - Unix system
    - Break into single account, then exploit local vulnerability or mount a "stepping stones" attack
  - Calling cards
  - Building
- To prevent an attacker from breaking into <u>any</u> account on <u>any</u> system

Also Need

- Usability!
  - Remember password?
  - Have to bring physical object with us all the time?
- Denial of service
  - Stolen wallet
  - Try to authenticate as you until your account becomes locked
  - What about a military or other mission critical scenario
    - Lock <u>all</u> accounts - system unusable

Password-Based Authentication

- User has a secret password.
  System checks it to authenticate the user.
  - May be vulnerable to eavesdropping when password is communicated from user to system
- How is the password stored?
- How does the system check the password?
- How easy is it to remember the password?
- How easy is it to guess the password?
  - Easy-to-remember passwords tend to be easy to guess
  - Password file is difficult to keep secret

Common usage modes

Amazon = t0p53cr37

UWNetID = f0084r#1

Bank = a2z@m0$;

Image from http://www.interactivetools.com/staff/dave/damons_office/

Common usage modes

- Write down passwords
- Share passwords with others
- Use a single password across multiple sites
  - Amazon.com and Bank of America?
  - UW CSE machines and MySpace?
- Use easy to remember passwords
  - Favorite <something>?
  - Name + <number>?
- Other "authentication" questions
  - Mother's maiden name?

Some anecdotes [Dhamija and Perrig]

- Users taught how to make secure passwords, but chose not to do so

- Reasons:
  - Awkward or difficult
  - No accountability
  - Did not feel that it was important

Social Engineering

- “Hi, I’m the CEO’s assistant. I need you to reset his password right away. He’s stuck in an airport and can’t log in! He lost the paper that he wrote the password on.
- “What do you mean you can’t do it!? Do you really want me to tell him that you’re preventing him from closing this major deal?
- “Great! That’s really helpful. You have no idea how important this is. Please set the password to ABCDEFG. He’ll reset it again himself right away.
- “Thanks!”

University of Sydney Study [Greening '96]

- 336 CS students emailed message asking them to supply their password
  - Pretext: in order to "validate" the password database after a suspected break-in
- 138 students returned their password
- 30 returned invalid password
- 200 changed their password
- (Not disjoint)

- Still, 138 is a lot!

Awkward

- How many times do you have to enter your password before it actually works?
  - Sometimes quite a few for me! (Unless I type extra slowly.)
- Interrupts normal activity
  - Do you lock your computer when you leave for 5 minutes?
  - Do you have to enter a password when your computer first boots? (Sometimes it's an option.)
- And <u>memorability</u> is an issue!

Memorability [Anderson]

- Hard to remember many PINs and passwords
- One bank had this idea
  - If pin is 2256, write your favorite 4-letter word in this grid
  - Then put random letters everywhere else

1	2	3	4	5	6	7	8	9	0
	b								
	l								
				u					
					e				

Memorability [Anderson]

- Problem!
- Normally 10000 choices for the PIN --- hard to guess on the first try
- Now, only a few dozen possible English words --- easy to guess on first try!

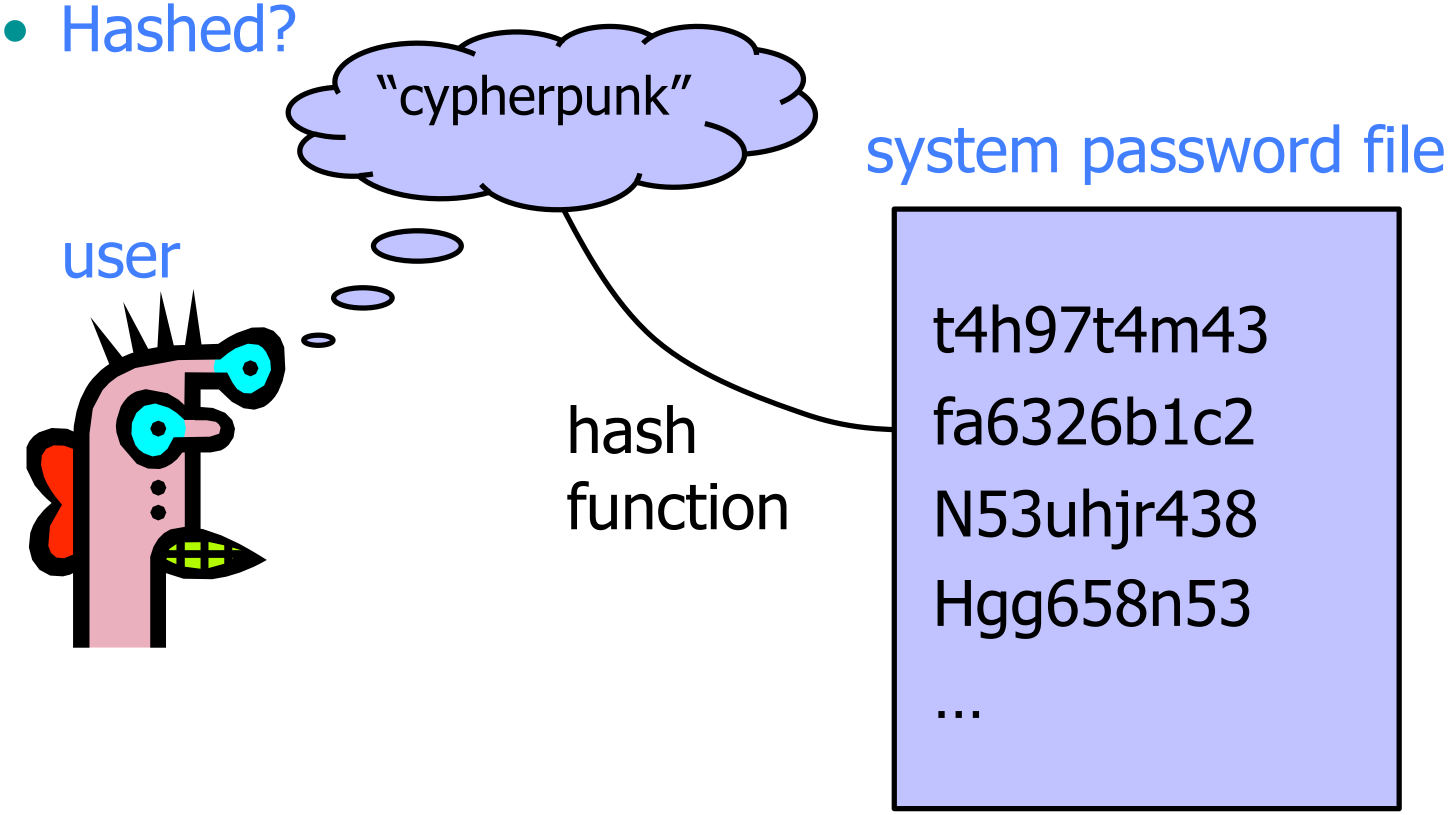
1	2	3	4	5	6	7	8	9	0
	b								
	l								
				u					
					e				

UNIX-Style Passwords

- How should we store passwords on a server?
  - In cleartext?
  - Encrypted?
  - Hashed?

UNIX-Style Passwords

- How should we store passwords on a server?
  - In cleartext?
  - Encrypted?
  - Hashed?



Password Hashing

- Instead of user password, store H(password)
- When user enters password, compute its hash and compare with entry in password file
  - System does not store actual passwords!
  - System itself can't easily go from hash to password
    - Which would be possible if the passwords were encrypted
- Hash function H must have some properties
  - One-way: given H(password), hard to find password
    - No known algorithm better than trial and error
    - It should even be hard to find any pair p1,p2 s.t. H(p1)=H(p2)

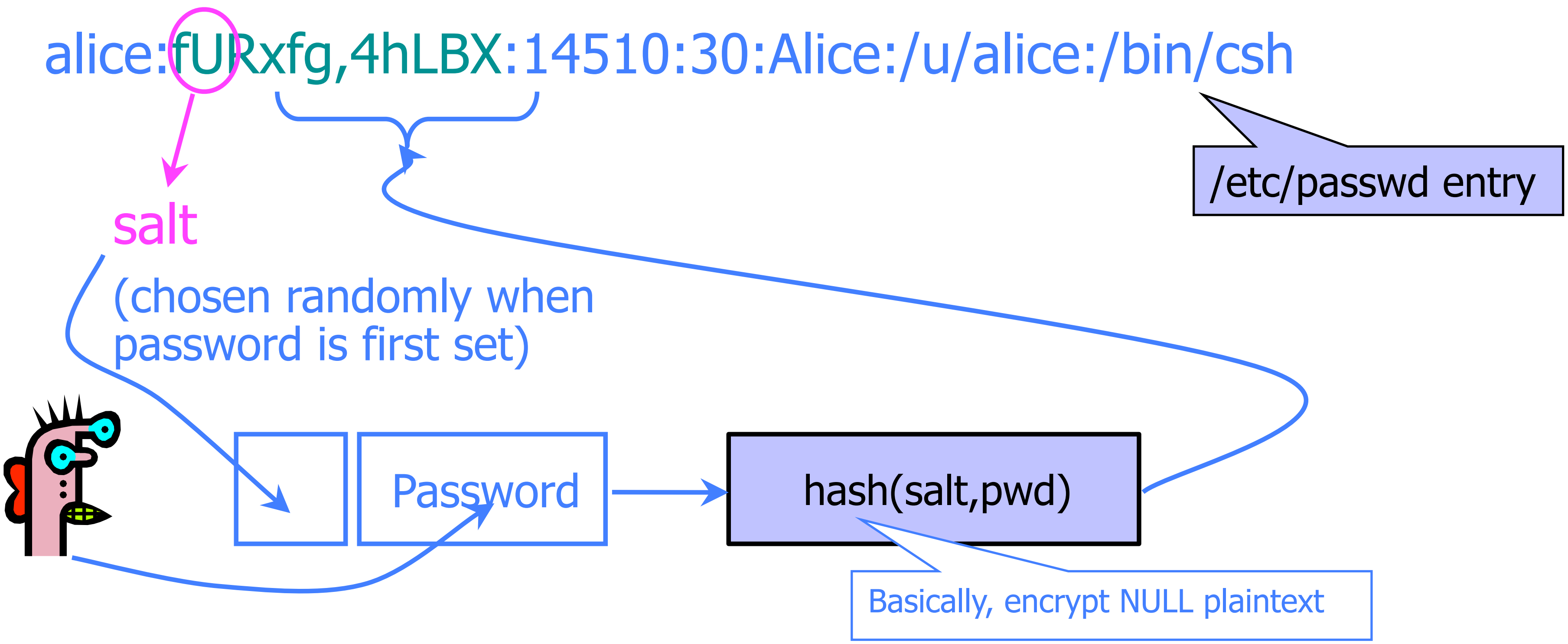
UNIX Password System

- Uses DES encryption as if it were a hash function
  - Encrypt NULL string using password as the key
    - Truncates passwords to 8 characters!
  - Artificial slowdown: run DES 25 times
    - Why 25 times? Slowdowns like these are important in practice!
  - Can instruct modern UNIXes to use MD5 hash function
- Problem: passwords are not truly random
  - With 52 upper- and lower-case letters, 10 digits and 32 punctuation symbols, there are $94^8 \approx$ 6 quadrillion possible 8-character passwords (around 2^{52})
  - Humans like to use dictionary words, human and pet names $\approx$ 1 million common passwords

Dictionary Attack

- Password file /etc/passwd is world-readable
  - Contains user IDs and group IDs which are used by many system programs
- Dictionary attack is possible because many passwords come from a small dictionary
  - Attacker can compute H(word) for every word in the dictionary and see if the result is in the password file
  - With 1,000,000-word dictionary and assuming 10 guesses per second, brute-force online attack takes 50,000 seconds (14 hours) on average
    - This is very conservative. Offline attack is much faster!
    - As described, could just create dictionary of word-->H(word) once!!

Salt



- Users with the same password have <u>different</u> entries in the password file
- Dictionary attack is still possible!

Advantages of Salting

- Without salt, attacker can pre-compute hashes of all dictionary words once for <u>all</u> password entries
  - Same hash function on all UNIX machines
  - Identical passwords hash to identical values; one table of hash values can be used for all password files
- With salt, attacker must compute hashes of all dictionary words once for <u>each</u> password entry
  - With 12-bit random salt, same password can hash to 2^{12} different hash values
  - Attacker must try all dictionary words for each salt value in the password file

Shadow Passwords

alice:x:14510:30:Vitaly:/u/alice:/bin/csh

/etc/passwd entry

Hashed password is not
stored in a world-readable file

- Store hashed passwords in /etc/shadow file which is only readable by system administrator (root)
- Add expiration dates for passwords
- Early Shadow implementations on Linux called the login program which had a buffer overflow!

Other Password Issues

- Keystroke loggers
  - Hardware
  - Software / Spyware
- Shoulder surfing
  - It's happened to me!
- Online vs offline attacks
  - Online: slower, easier to respond
- Multi-site authentication
  - Share passwords?

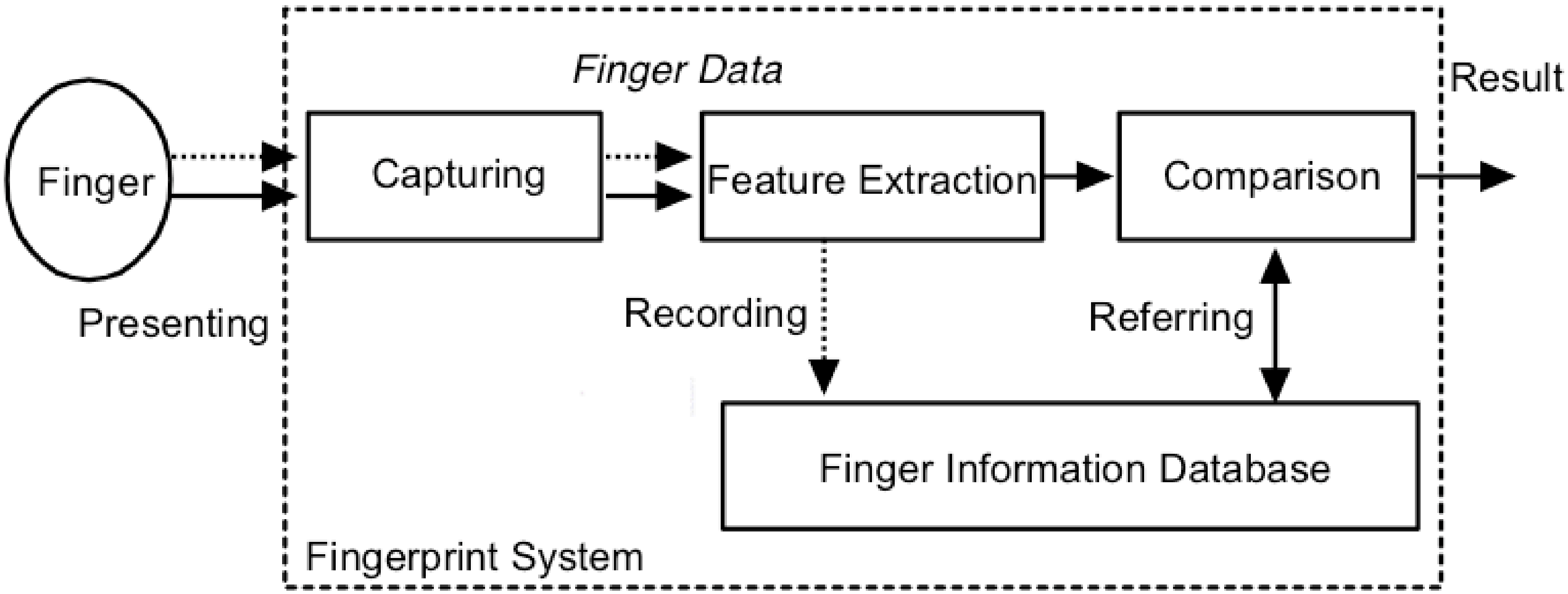
Implementation Attacks

- Smartcard had a PIN-retry counter
  - By monitoring power line, can detect if PIN incorrect
  - If so, reset quickly
  - Can now circumvent PIN-retry counter
- Timing attack in TENEX password verification system

What About Biometrics?

- Authentication: What you are
- Unique identifying characteristics to authenticate user or create credentials
  - Biological and physiological: Fingerprints, iris scan
  - Behaviors characteristics - how perform actions: Handwriting, typing, gait
- Advantages:
  - Nothing to remember
  - Passive
  - Can’t share (generally)
  - With perfect accuracy, could be fairly unique

Overview [Matsumoto]



Tsutomu Matsumoto's image, from http://web.mit.edu/6.857/OldStuff/Fall03/ref/gummy-slides.pdf

Dashed lines for enrollment; solid for verification or identification

Biometric Error Rates (Non-Adversarial)

- "Fraud rate" vs. "insult rate"
  - Fraud = system incorrectly accepts (false accept)
  - Insult = system rejects valid user (false reject)
- Increasing acceptance threshold increases fraud rate, decreases insult rate
  - Pick a threshold so that fraud rate = insult rate
- For biometrics, U.K. banks set target fraud rate of 1%, insult rate of 0.01% [Ross Anderson]
  - Common signature recognition systems achieve equal error rates around 1% - not good enough!

Biometrics

- Face recognition (by a computer algorithm)
  - Error rates up to 20%, given reasonable variations in lighting, viewpoint and expression
- Fingerprints
  - Traditional method for identification
  - 1911: first US conviction on fingerprint evidence
  - U.K. traditionally requires 16-point match
    - Probability of false match is 1 in 10 billion
    - No successful challenges until 2000
  - Fingerprint damage impairs recognition
    - Ross Anderson’s scar crashes FBI scanner

Other Biometrics

- Iris scanning
  - Irises are very random, but stable through life
    - Different between the two eyes of the same individual
  - 256-byte iris code based on concentric rings between the pupil and the outside of the iris
  - Equal error rate better than 1 in a million
  - Best biometric mechanism currently known
- Hand geometry
  - Used in nuclear premises entry control, INSPASS (discontinued in 2002)

Other Biometrics

- Vein
  - Pattern on back of hand
- Handwriting
- Typing
  - Timings for character sequences
- Gait
- DNA

Issues with Biometrics

- Private, but not secret
  - Maybe encoded on the back of an ID card?
  - Maybe encoded on your glass, door handle, ...
  - Sharing between multiple systems?
- Revocation is difficult (impossible?)
  - Sorry, your iris has been compromised, please create a new one...
- Physically identifying
  - Soda machine to cross-reference fingerprint with DMV?

Issues with Biometrics

- Criminal gives an inexperienced policeman fingerprints in the wrong order
  - Record not found; gets off as a first-time offender
- Can be attacked using recordings
  - Ross Anderson: in countries where fingerprints are used to pay pensions, there are persistent tales of "Granny's finger in the pickle jar" being the most valuable property she bequeathed to her family
- Birthday paradox
  - With false accept rate of 1 in a million, probability of false match is above 50% with only 1609 samples

Issues with Biometrics

- Anecdotally, car jackings went up when it became harder to steal cars without the key
- But what if you need your fingerprint to start your car?
  - Stealing cars becomes harder
  - So what would the car thieves have to do?

Risks of Biometrics

BBC NEWS

OPEN The News in 2 minutes

News services
Your news when
want it

News Front Page
Africa
Americas
Asia-Pacific
Europe
Middle East
South Asia
UK
Business
Health
Science/Nature
Technology
Entertainment

Last Updated: Thursday, 31 March, 2005, 10:37 GMT 11:37 UK

E-mail this to a friend Printable version

Malaysia car thieves steal finger

By Jonathan Kent
BBC News, Kuala Lumpur

Police in Malaysia are hunting for members of a violent gang who chopped off a car owner's finger to get round the vehicle's hi-tech security system.

The car, a Mercedes S-class, was protected by a fingerprint recognition system.

Accountant K Kumaran's ordeal began when he was run down by four men in a small car as he was about to get into his Mercedes in a Kuala Lumpur suburb.

SEE ALSO:
- Malaysia to act
pirates
16 Mar 05 | As

RELATED INTER
- Malaysian police

The BBC is not r
for the content o
internet sites

TOP ASIA-PACIF STORIES
- Australians war
cuts
- Taiwan campus

http://news.bbc.co.uk/2/hi/asia-pacific/4396831.stm

Biometric Error Rates (Adversarial)

- Want to minimize “fraud” and “insult” rate
  - “Easy” to test probability of accidental misidentification (fraud)
  - But what about adversarial fraud
    - Besides stolen fingers
- An adversary might try to steal the biometric information
  - Malicious fingerprint reader
    - Consider when biometric is used to derive a cryptographic key
  - Residual fingerprint on a glass

Voluntary: Making a Mold

[Matsumoto]

Put the plastic into hot water to soften it.

Press a live finger against it.

The mold

It takes around 10 minutes.

http://web.mit.edu/6.857/OldStuff/Fall03/ref/gummy-slides.pdf

Voluntary: Making a Finger

[Matsumoto]

Pour the liquid into the mold.

Put it into a refrigerator to cool.

The gummy finger

It takes around 10 minutes.

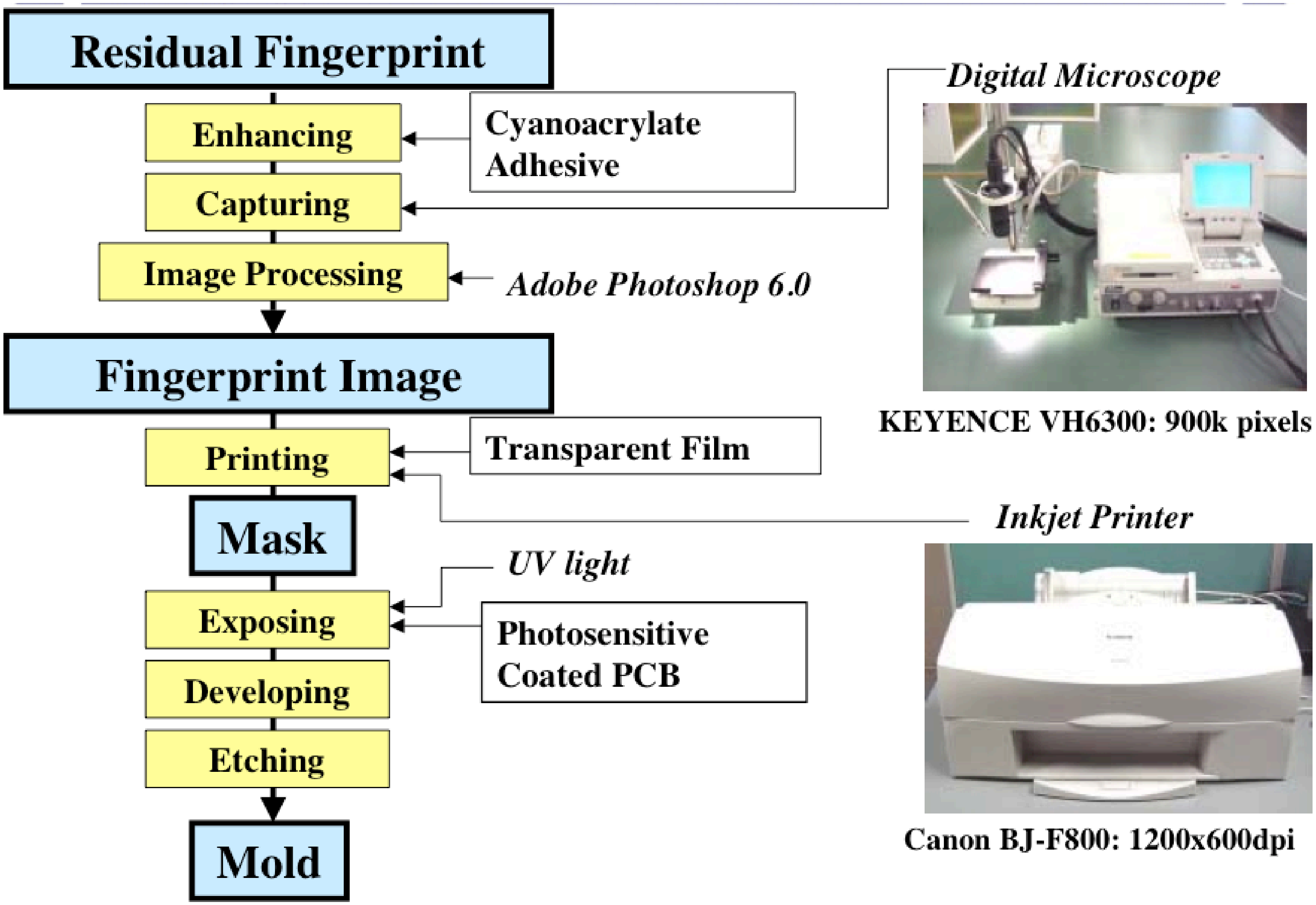
http://web.mit.edu/6.857/OldStuff/Fall03/ref/gummy-slides.pdf

Voluntary

- Only costs a few dollars
- We will (hopefully!) try this later in the course
  - I’ve ordered some supplies
  - But they’re not here yet...

Involuntary

[Matsumoto]

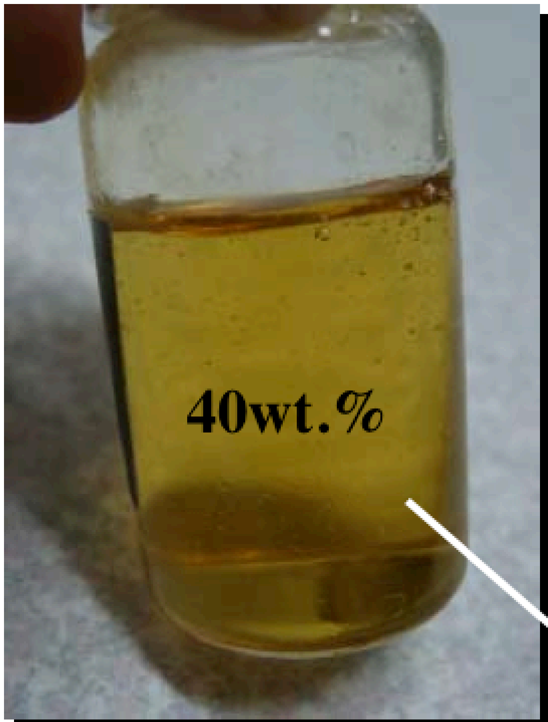


http://web.mit.edu/6.857/OldStuff/Fall03/ref/gummy-slides.pdf

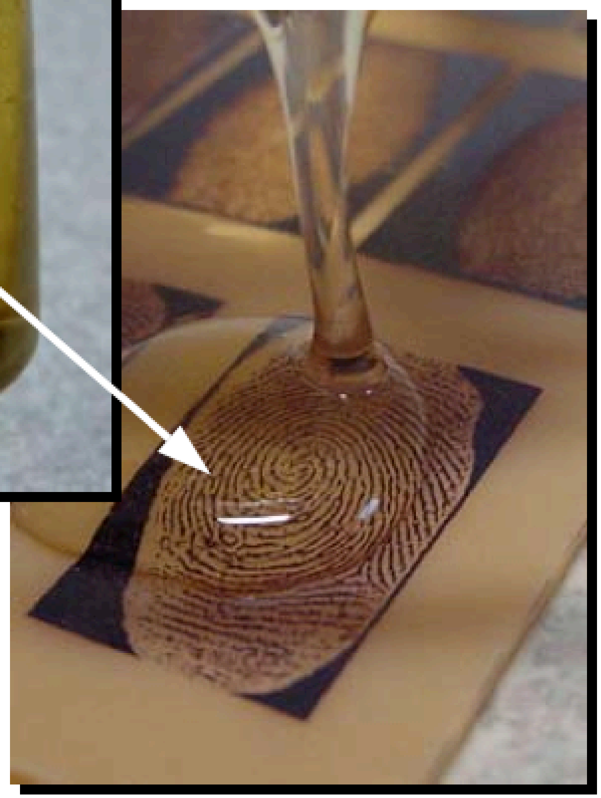
Involuntary

[Matsumoto]

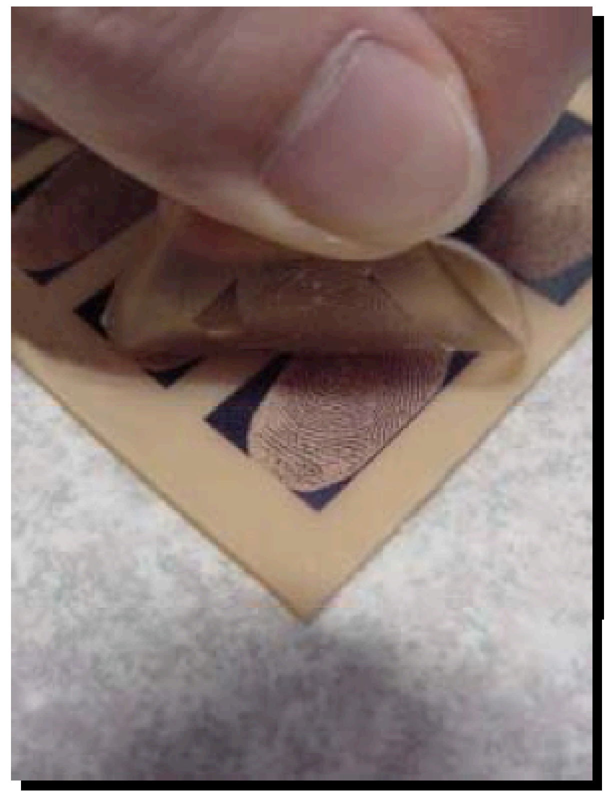
Gelatin Liquid



Drip the liquid onto the mold.



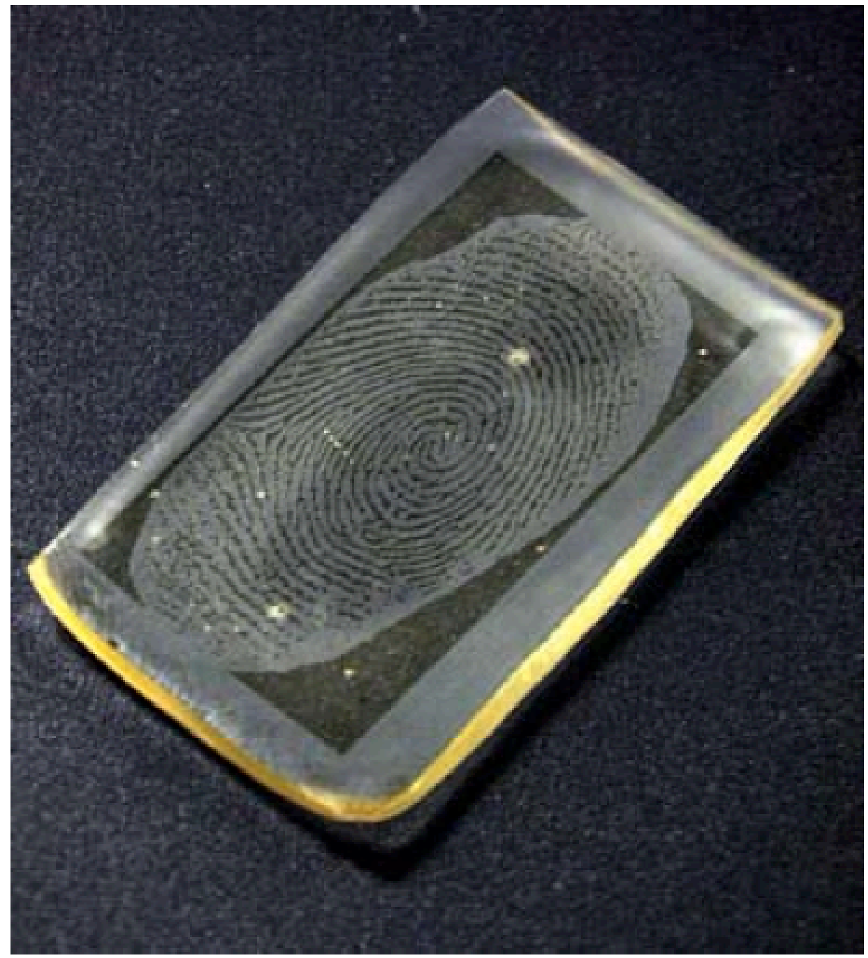
Put this mold into a refrigerator to cool, and then peel carefully.



http://web.mit.edu/6.857/OldStuff/Fall03/ref/gummy-slides.pdf

Involuntary

[Matsumoto]

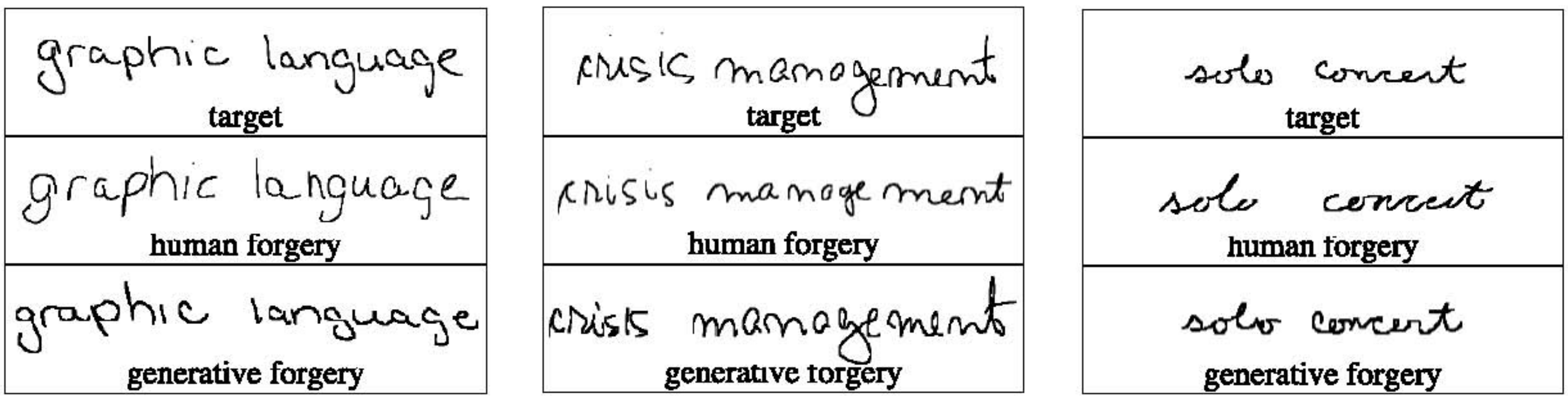


http://web.mit.edu/6.857/OldStuff/Fall03/ref/gummy-slides.pdf

Authentication by Handwriting

[Ballard, Monrose, Lopresti]

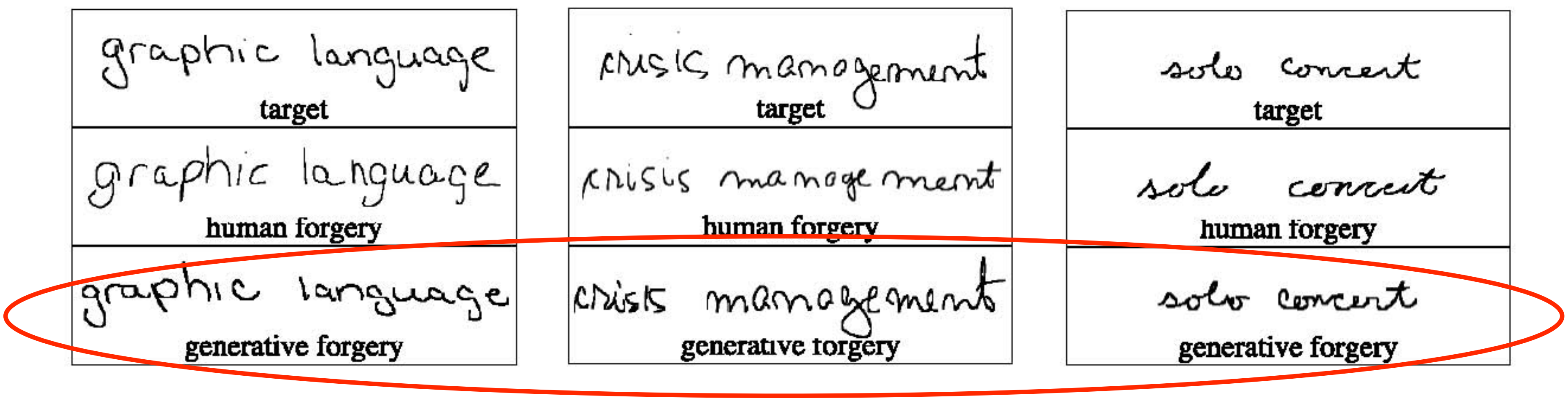
- Maybe a computer could also forge some biometrics



Authentication by Handwriting

[Ballard, Monrose, Lopresti]

- Maybe a computer could also forge some biometrics



Generated by computer algorithm trained on handwriting samples

Password Managers

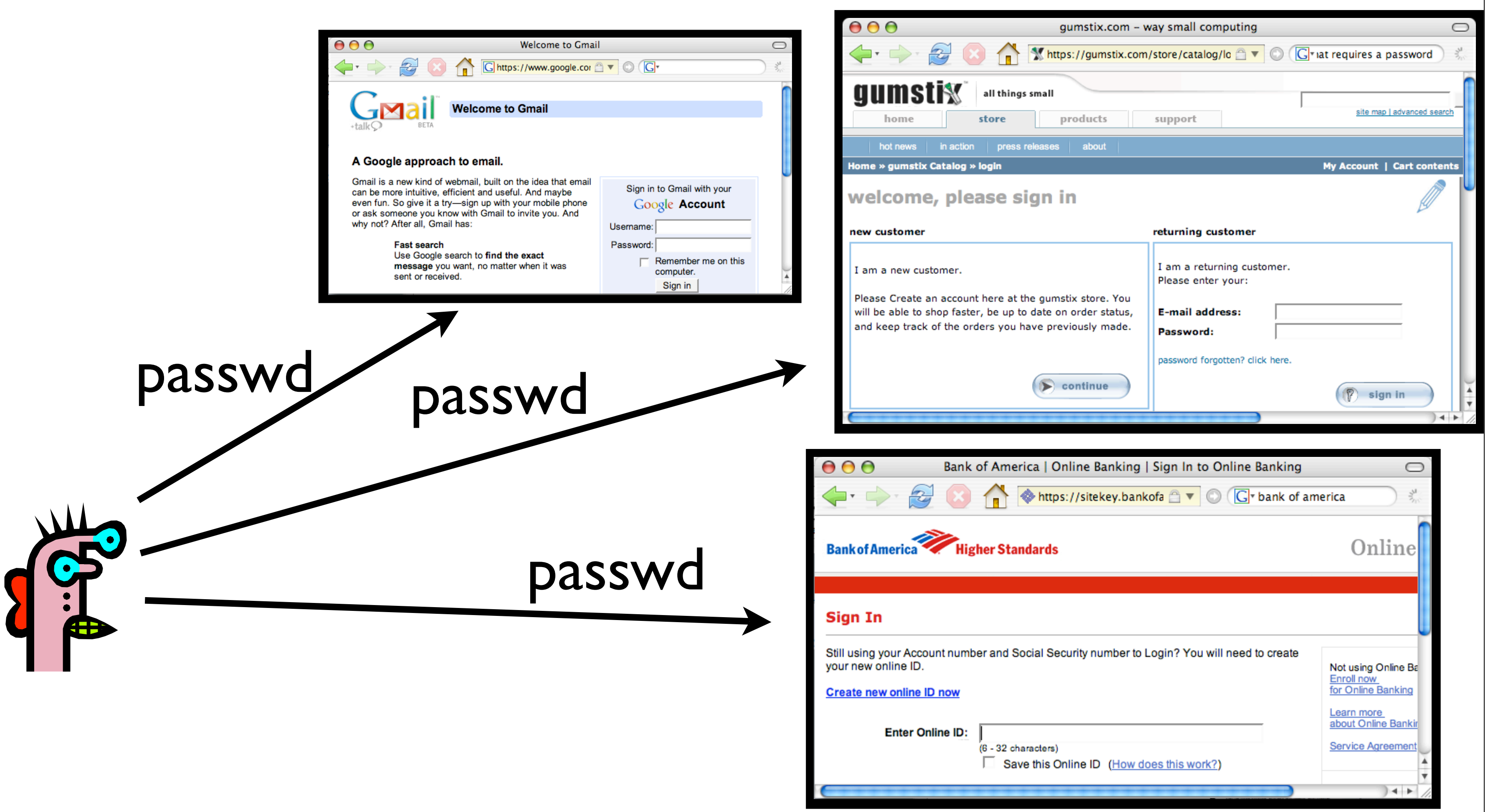
- Idea: Software application that will store and manage passwords for you.
  - You remember one password.
  - Each website sees a different password.
- Examples: PwdHash (Usenix Security 2005) and Password Multiplier (WWW 2005).

Key ideas

- User remembers a single password
- Password managers
  - On input: (1) the user's single password and (2) information about the website
  - Compute: Strong, site-specific password
- Goal: Avoid problems with passwords

The problem

Alice needs passwords for all the websites that she visits



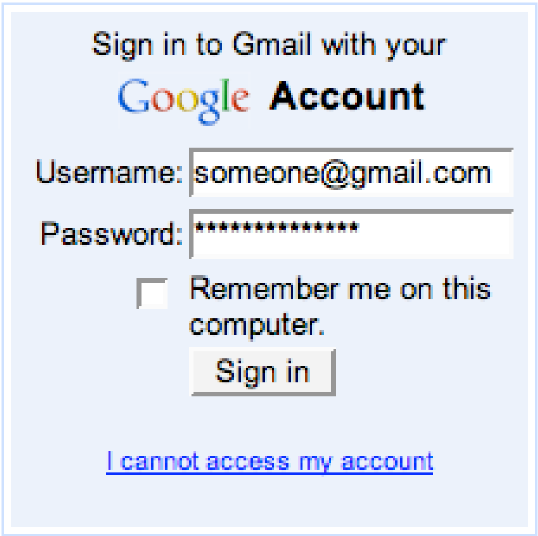
Possible solutions

- Easy to remember: Use same password on all websites. Use "weak" password.
  - Poor security (don't share password between bank website and small website)
- More secure: Use different, strong passwords on all websites.
  - Hard to remember, unless write down.

Alternate solution: Password managers

- Password managers handle creating and "remembering" strong passwords
- Potentially:
  - Easier for users
  - More secure
- Examples:
  - PwdHash (Usenix Security 2005)
  - Password Multiplier (WWW 2005)

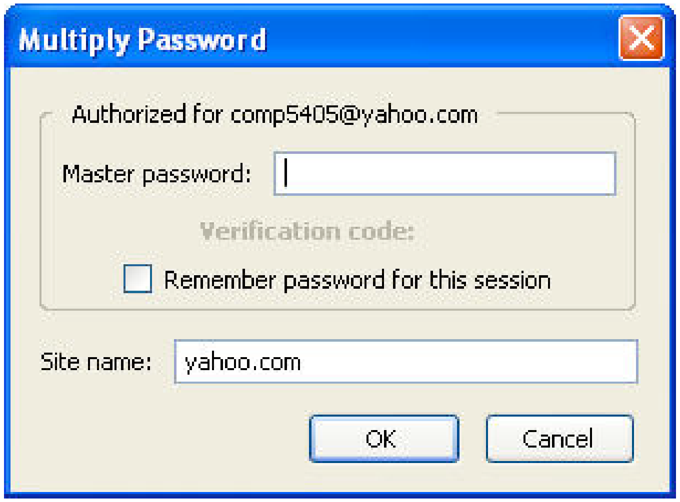
PwdHash



@@ in front of passwords to protect; or F2

sitePwd = func(pwd,domain)

Password Multiplier



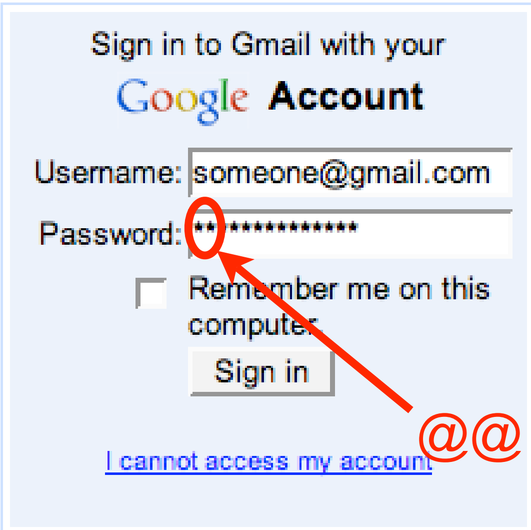
Active with Alt-P or double-click

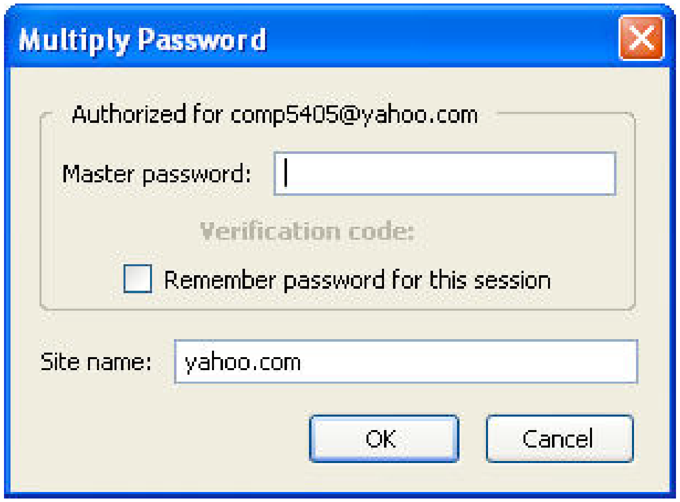
sitePwd = func(usrname, pwd, domain)

Both solutions target simplicity and transparency.

PwdHash

Password Multiplier





@@ in front of passwords to protect; or F2

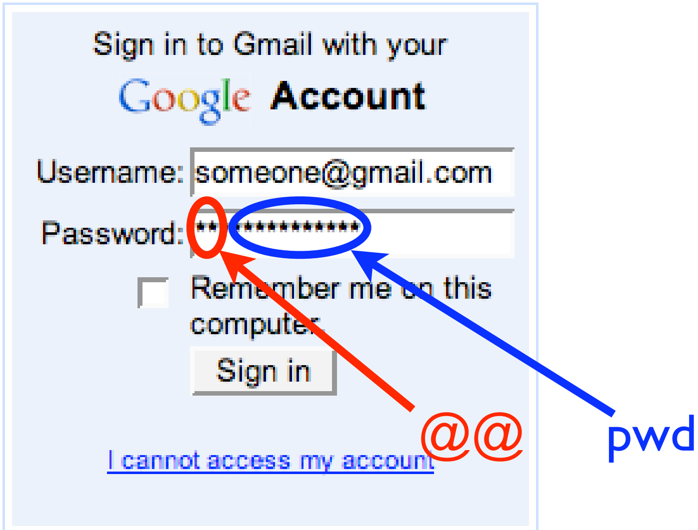
sitePwd = func(pwd,domain)

Active with Alt-P or double-click

sitePwd = func(usrname, pwd, domain)

Both solutions target simplicity and transparency.

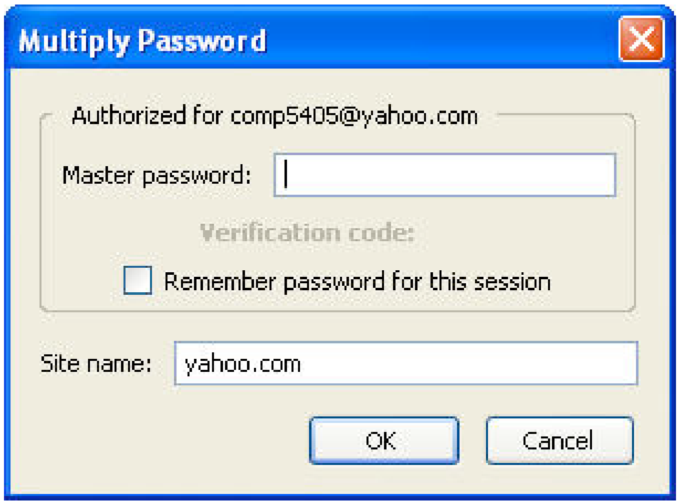
PwdHash



@@ in front of passwords to protect; or F2

sitePwd = func(pwd,domain)

Password Multiplier



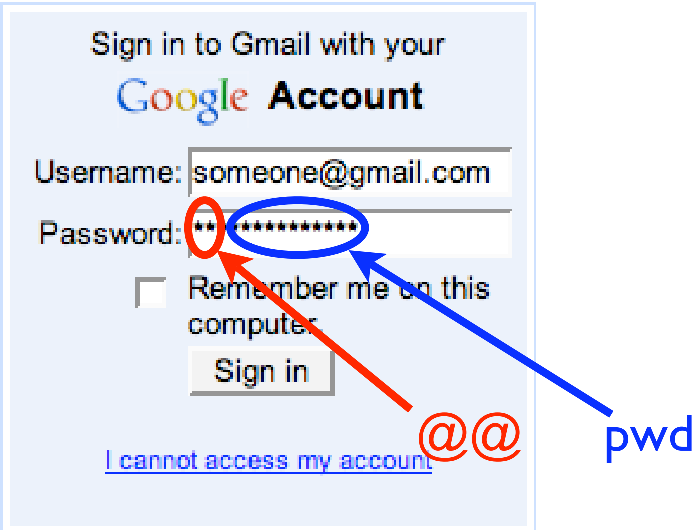
Active with Alt-P or double-click

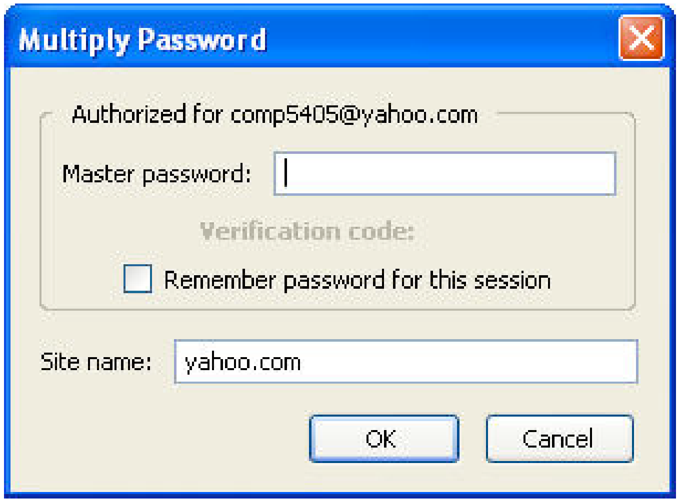
sitePwd = func(usrname, pwd, domain)

Both solutions target simplicity and transparency.

PwdHash

Password Multiplier





@@ in front of passwords to protect; or F2

sitePwd = func(pwd,domain)

Prevent phishing attacks

Active with Alt-P or double-click

sitePwd = func(usrname, pwd, domain)

Both solutions target simplicity and transparency.

Usenix 2006: Usabilty testing

- Are these programs usable? If not, what are the problems?
- Two main approaches for evaluating usability:
  - Usability inspection (no users)
    - Cognitive walk throughs
    - Heuristic evaluation
  - User study
    - Controlled experiments
    - Real usage

Usenix 2006: Usabilty testing

- Are these programs usable? If not, what are the problems?
- Two main approaches for evaluating usability:
  - Usability inspection (no users)
    - Cognitive walk throughs
    - Heuristic evaluation
  - User study
    - Controlled experiments
    - Real usage

This paper stresses need to observe real users

[Chiasson, van Oorschot, Biddle]

Study details

- 26 participants, across various backgrounds (4 technical)
- Five assigned tasks per plugin
- Data collection
    - Observational data (recording task outcomes, difficulties, misconceptions)
    - Questionnaire data (initial attitudes, opinions after tasks, post questionnaires)

[Chiasson, van Oorschot, Biddle]

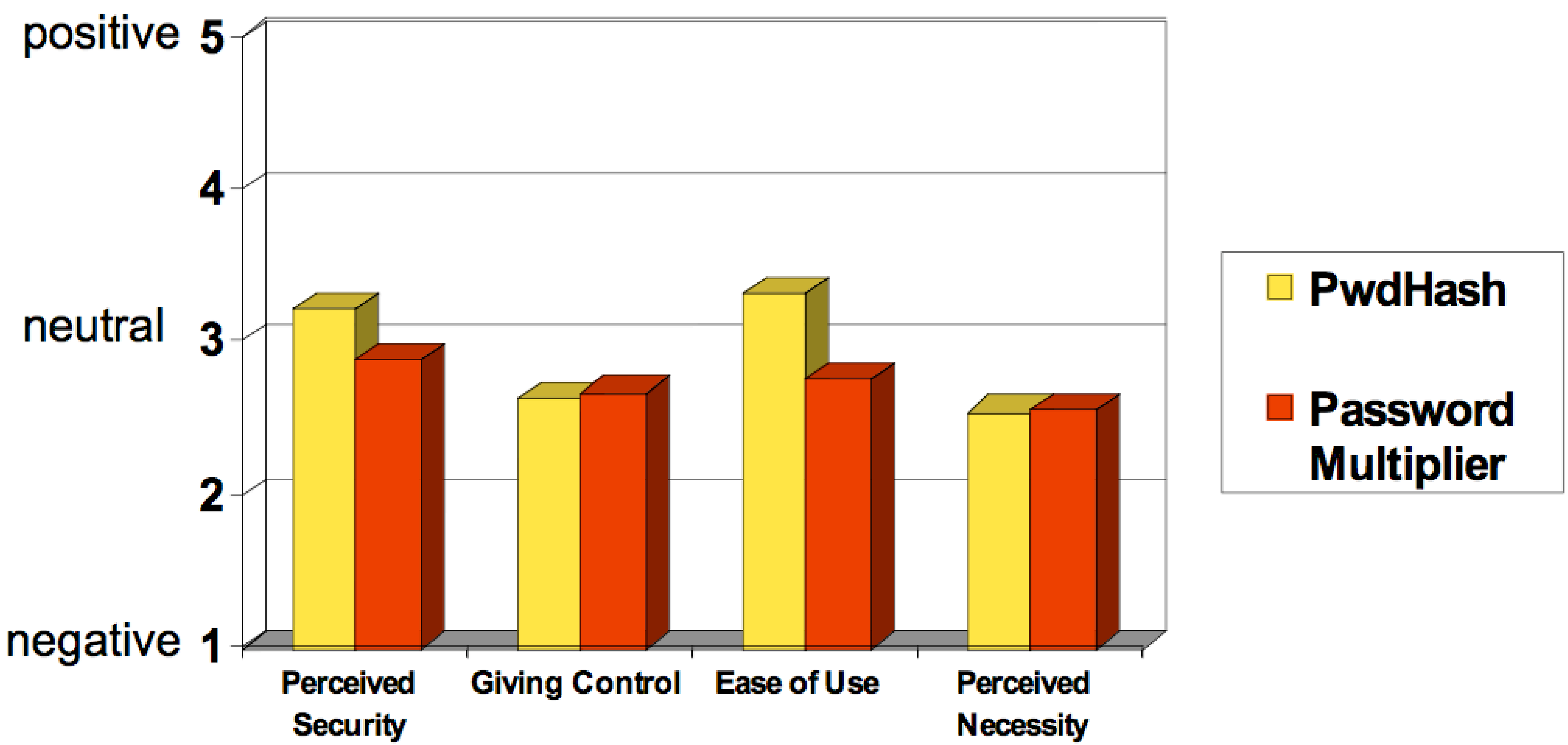
Task completion results

	Success	Potentially Causing Security Exposures			
		Dangerous Success	Failures		
			Failure	False Completion	Failed due to Previous
PwdHash					
Log In	48%	44%	8%	0%	N/A
Migrate Pwd	42%	35%	11%	11%	N/A
Remote Login	27%	42%	31%	0%	N/A
Update Pwd	19%	65%	8%	8%	N/A
Second Login	52%	28%	4%	0%	16%
Password Multiplier					
Log In	48%	44%	8%	0%	N/A
Migrate Pwd	16%	32%	28%	20%	N/A
Remote Login	N/A	N/A	N/A	N/A	N/A
Update Pwd	16%	4%	44%	28%	N/A
Second Login	16%	4%	16%	0%	16%

http://www.scs.carleton.ca/~schiasso/Chiasson_UsenixSecurity2006_PwdManagers.ppt

[Chiasson, van Oorschot, Biddle]

Questionnaire responses



http://www.scs.carleton.ca/~schiasso/Chiasson_UsenixSecurity2006_PwdManagers.ppt

Problem: Transparency

- Unclear to users whether actions successful or not.
  - Should be obvious when plugin activated.
  - Should be obvious when password protected.
- Users feel that they should be able to know their own password.

Problem: Mental model

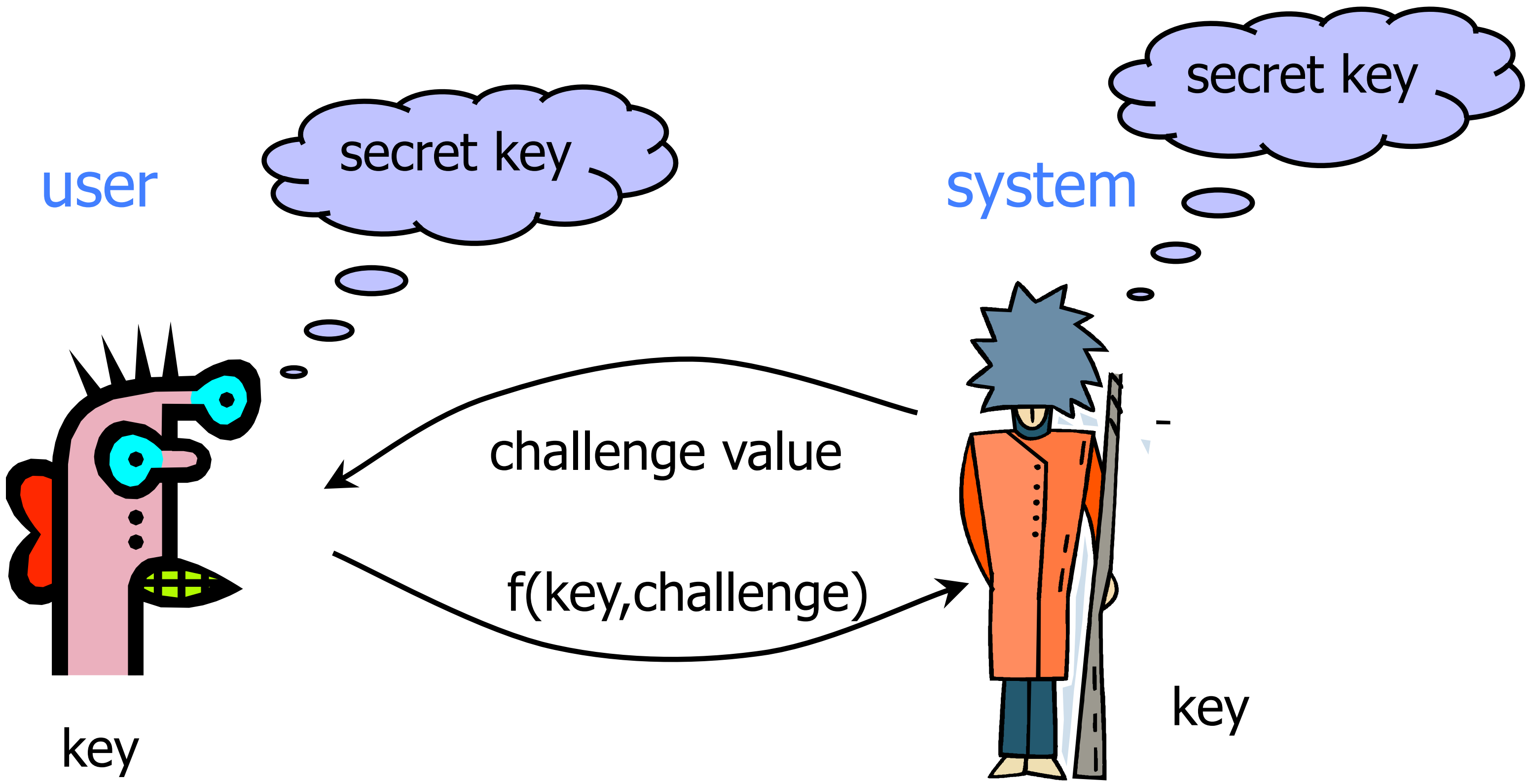
Users seemed to have misaligned mental models

- Not understand that one needs to put "@@" before *each* password to be protected.
- Think different passwords generated for each session.
- Think successful when were not.
- Not know to click in field before Alt-P.
- PwdHash: Think passwords unique to them.

When “nothing works”

- Tendency to try all passwords
  - A poor security choice.
  - May make the use of PwdHash or Password Multiplier *worse* than not using any password manager.
- Usability problem leads to security vulnerabilities.

Challenge-Response (Over Network)



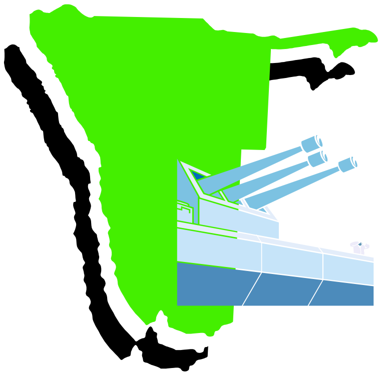
Why is this better than a password over a network?

Any problems remain?

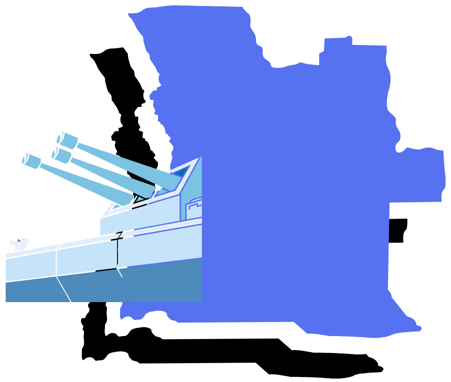
Challenge-Response Authentication

- User and system share a secret key
- Challenge: system presents user with some string
- Response: user computes response based on secret key and challenge
  - Secrecy: difficult to recover key from response
    - One-way hashing or symmetric encryption work well
  - Freshness: if challenge is fresh and unpredictable, attacker on the network cannot replay an old response
    - For example, use a fresh random number for each challenge
- Good for systems with pre-installed secret keys
  - Car keys; military friend-or-foe identification

MIG-in-the-Middle Attack [Ross Anderson]

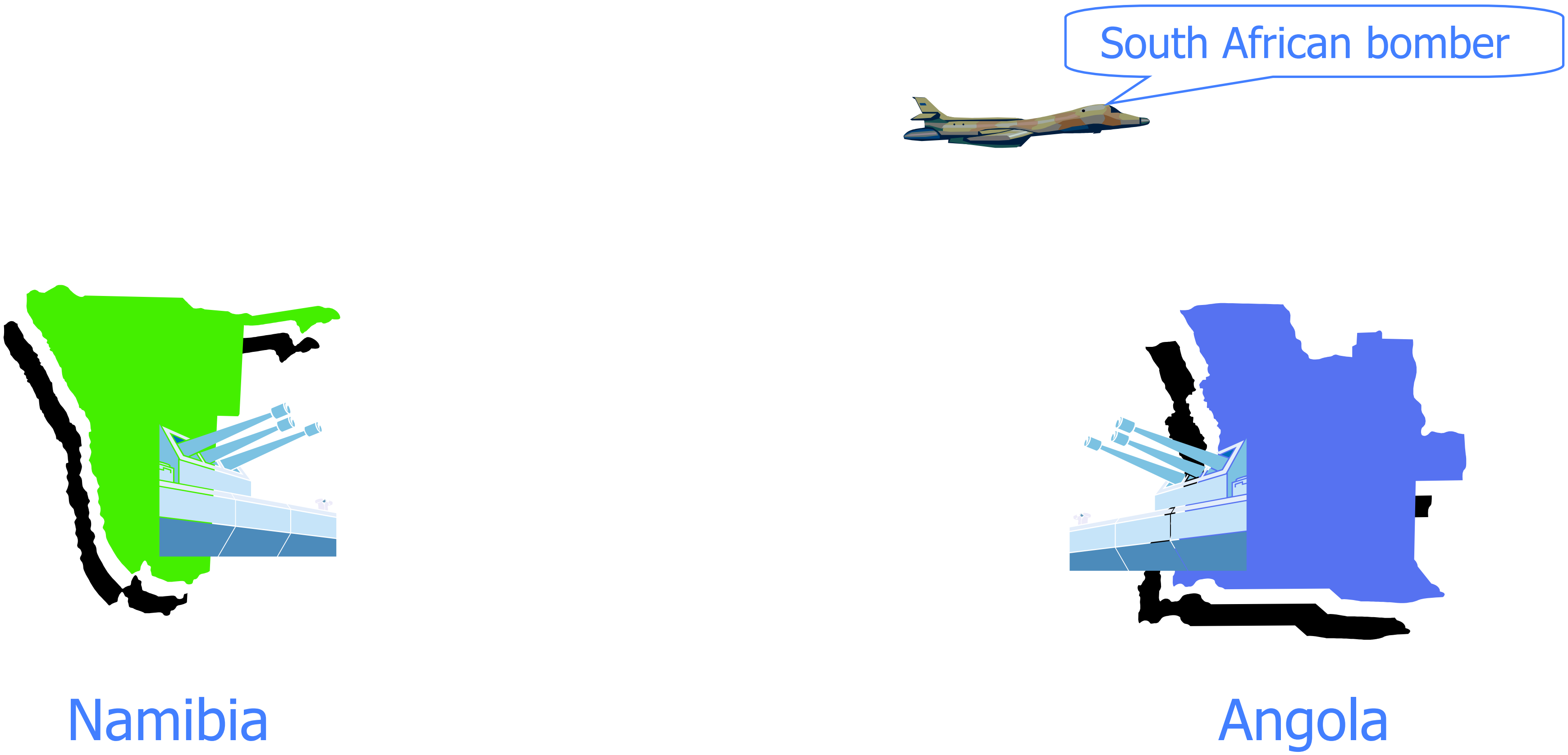


Namibia

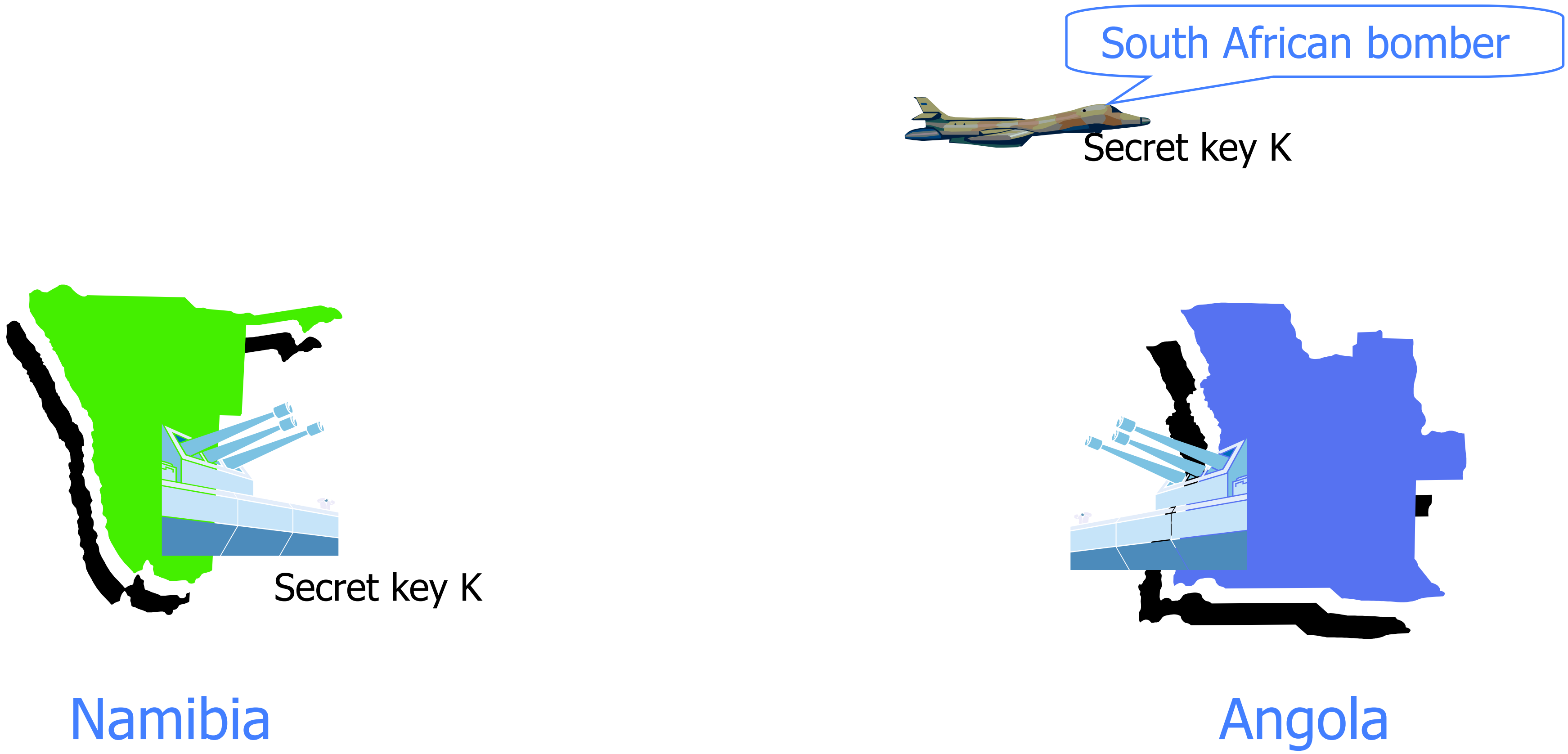


Angola

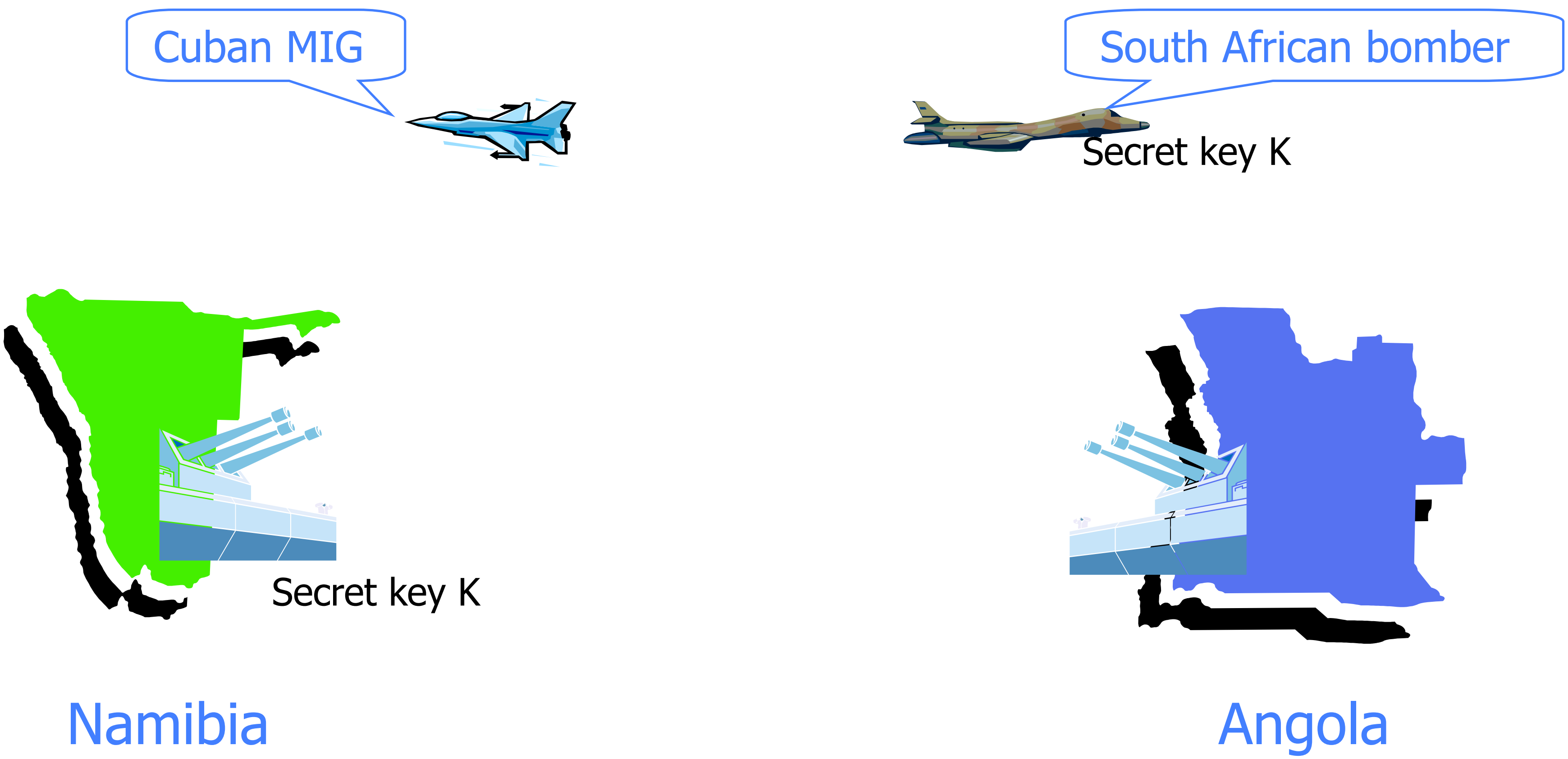
MIG-in-the-Middle Attack [Ross Anderson]



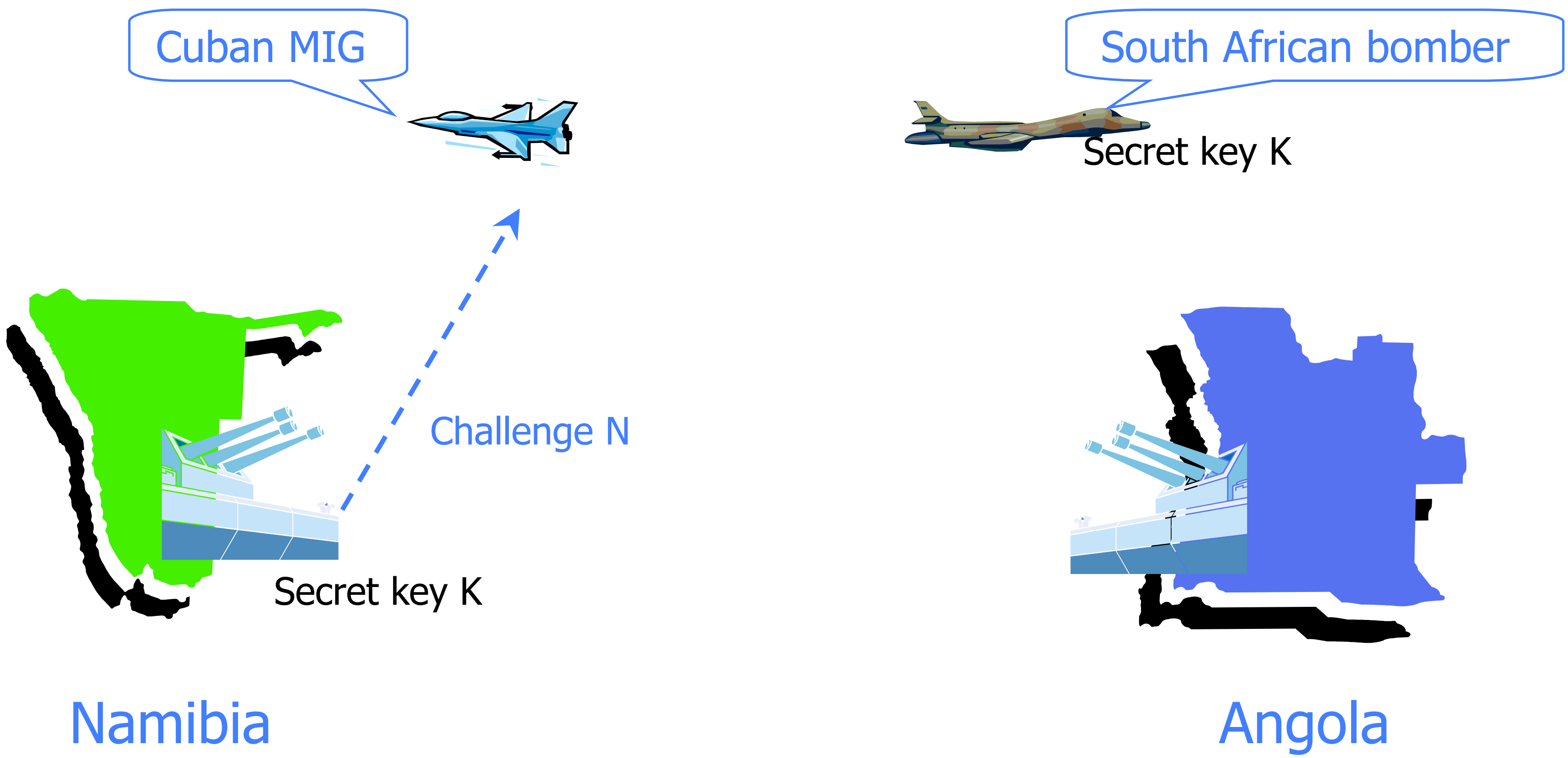
MIG-in-the-Middle Attack [Ross Anderson]



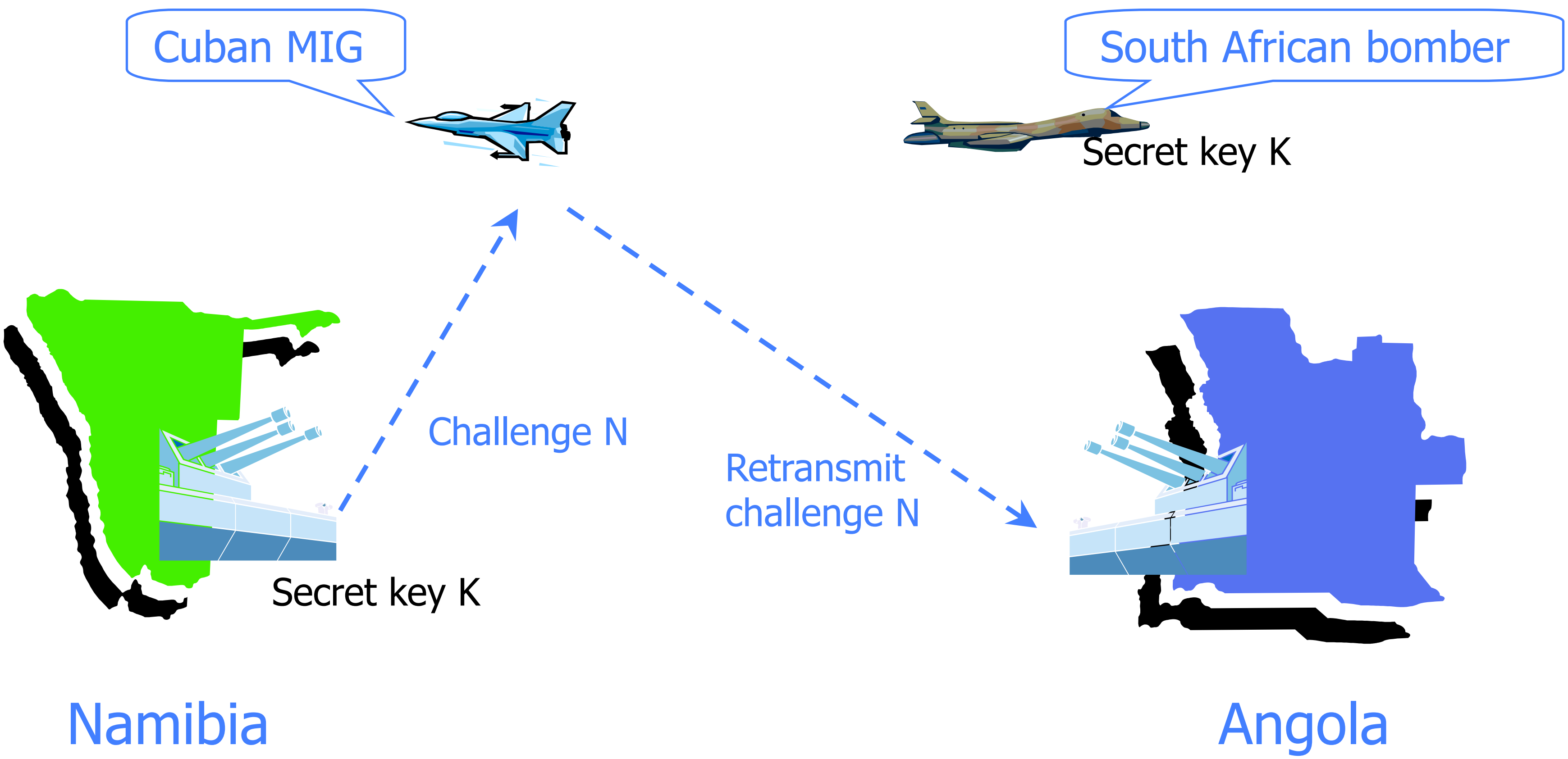
MIG-in-the-Middle Attack [Ross Anderson]



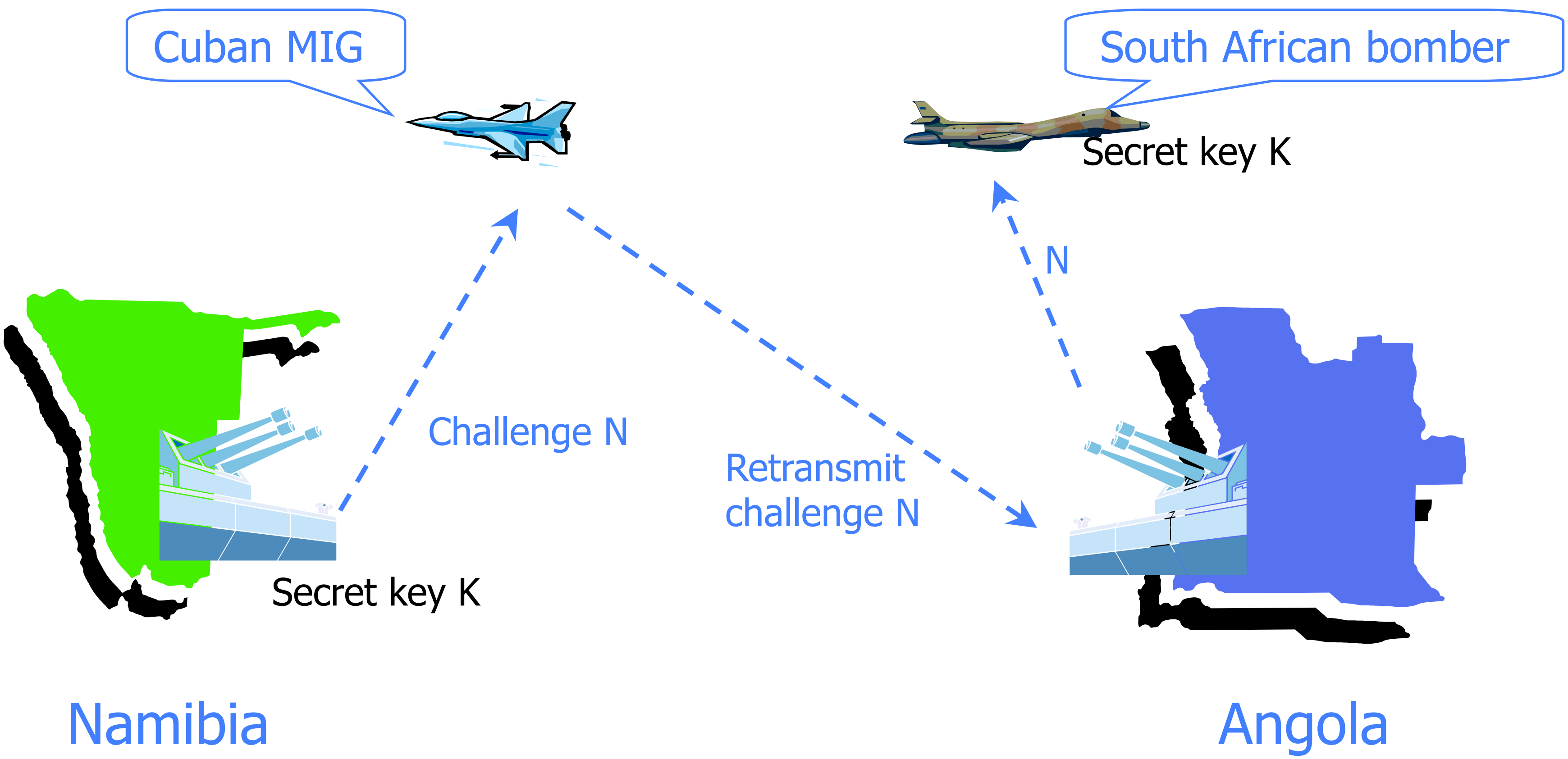
MIG-in-the-Middle Attack [Ross Anderson]



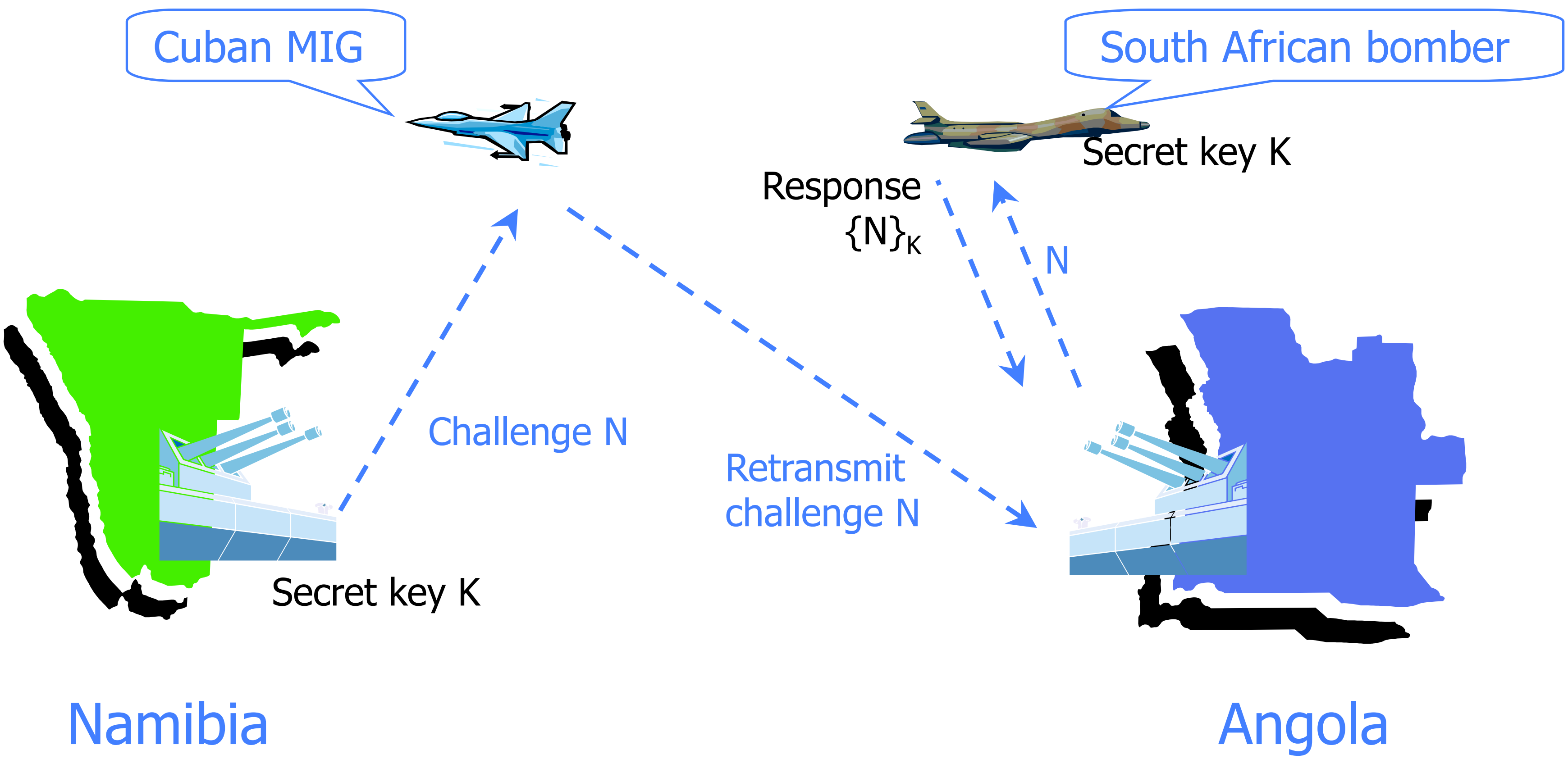
MIG-in-the-Middle Attack [Ross Anderson]



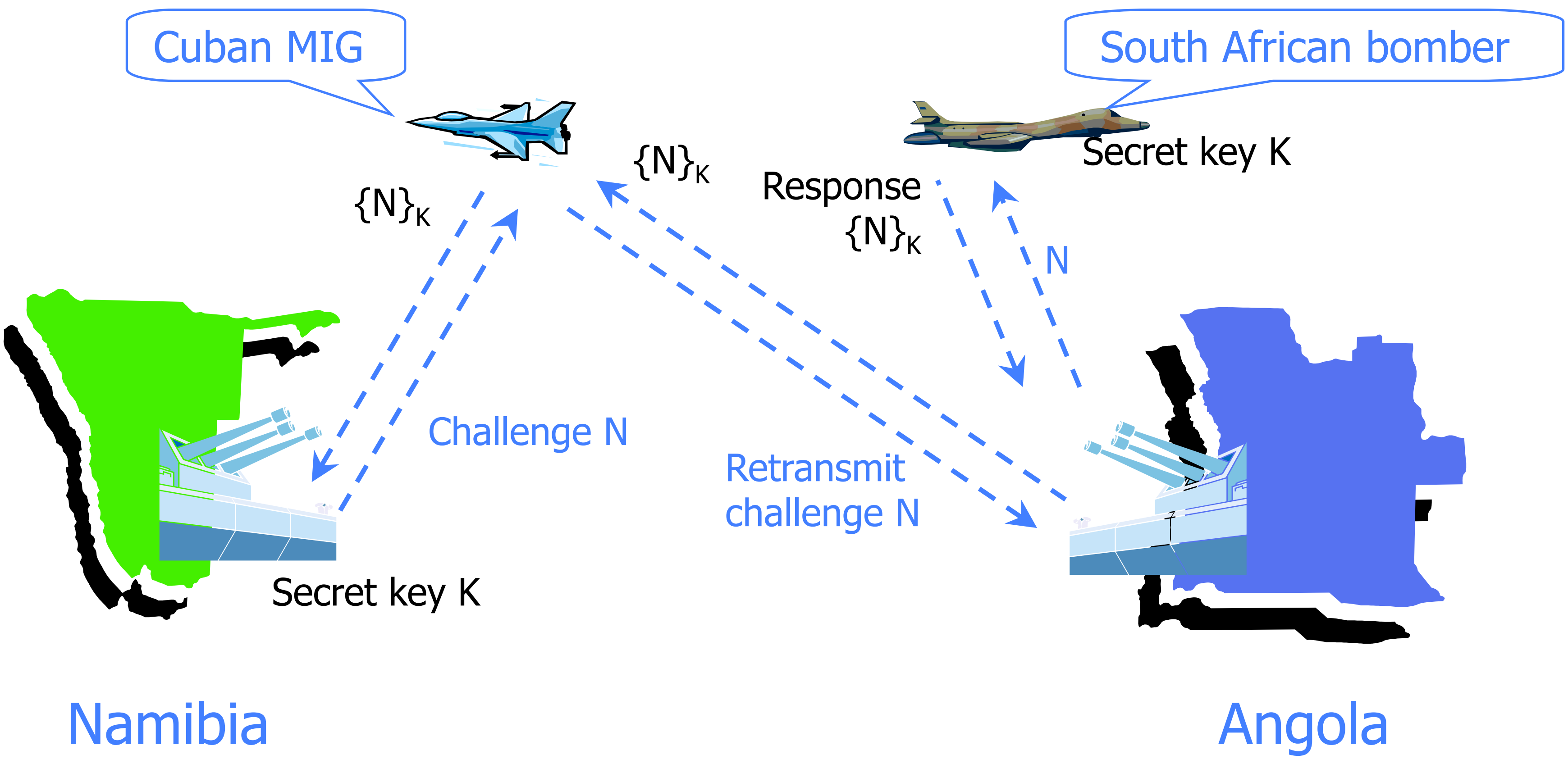
MIG-in-the-Middle Attack [Ross Anderson]



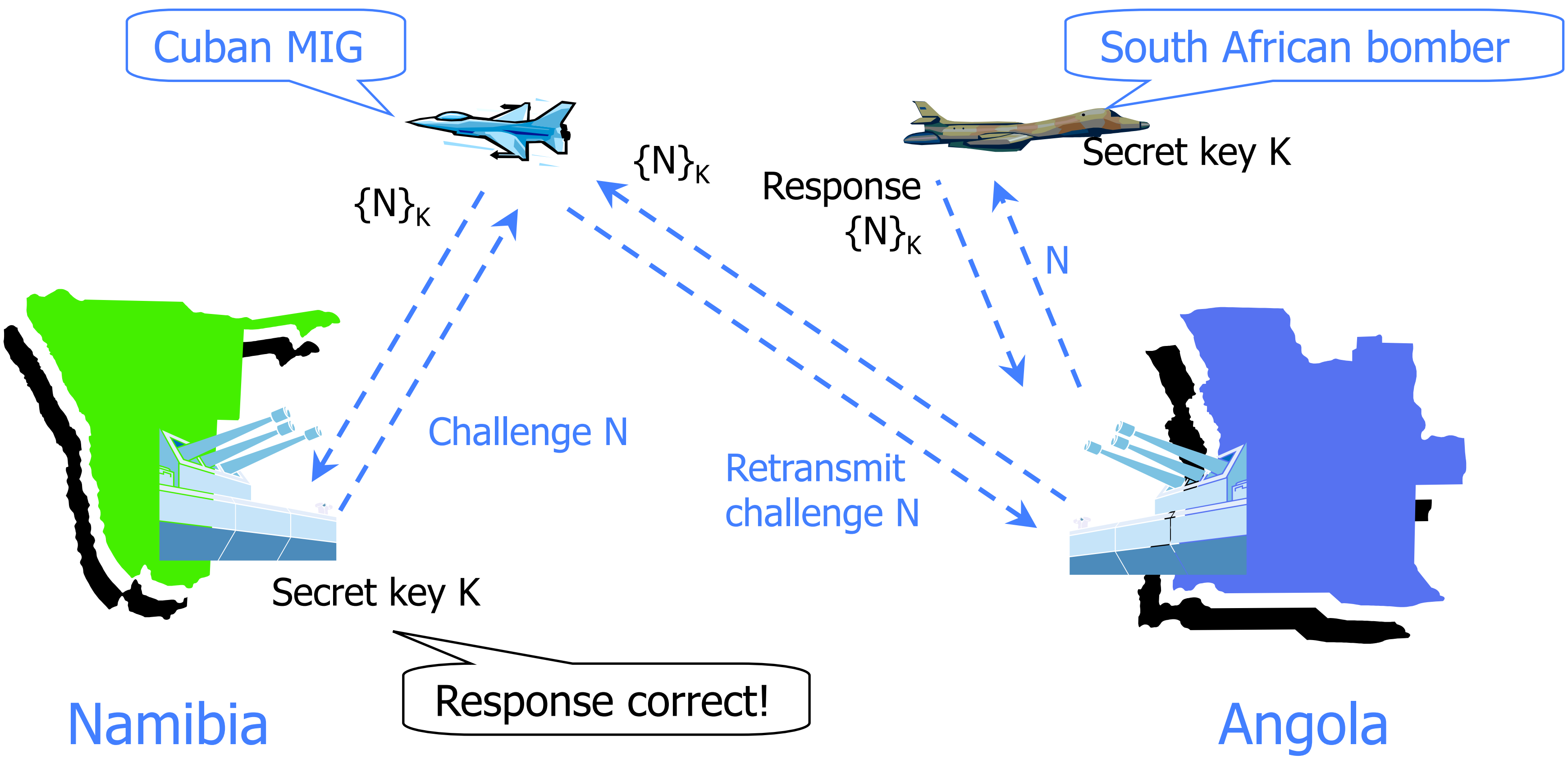
MIG-in-the-Middle Attack [Ross Anderson]



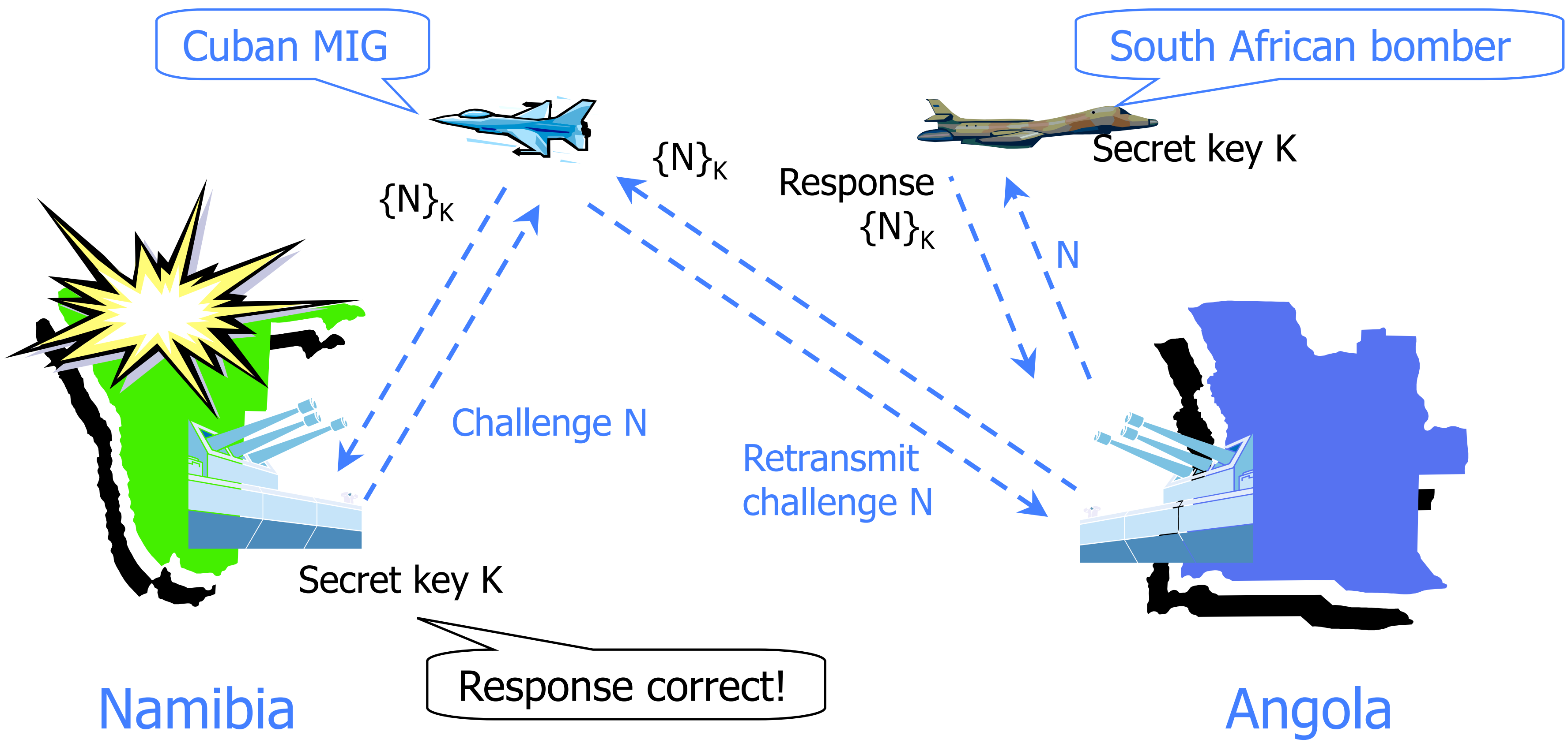
MIG-in-the-Middle Attack [Ross Anderson]



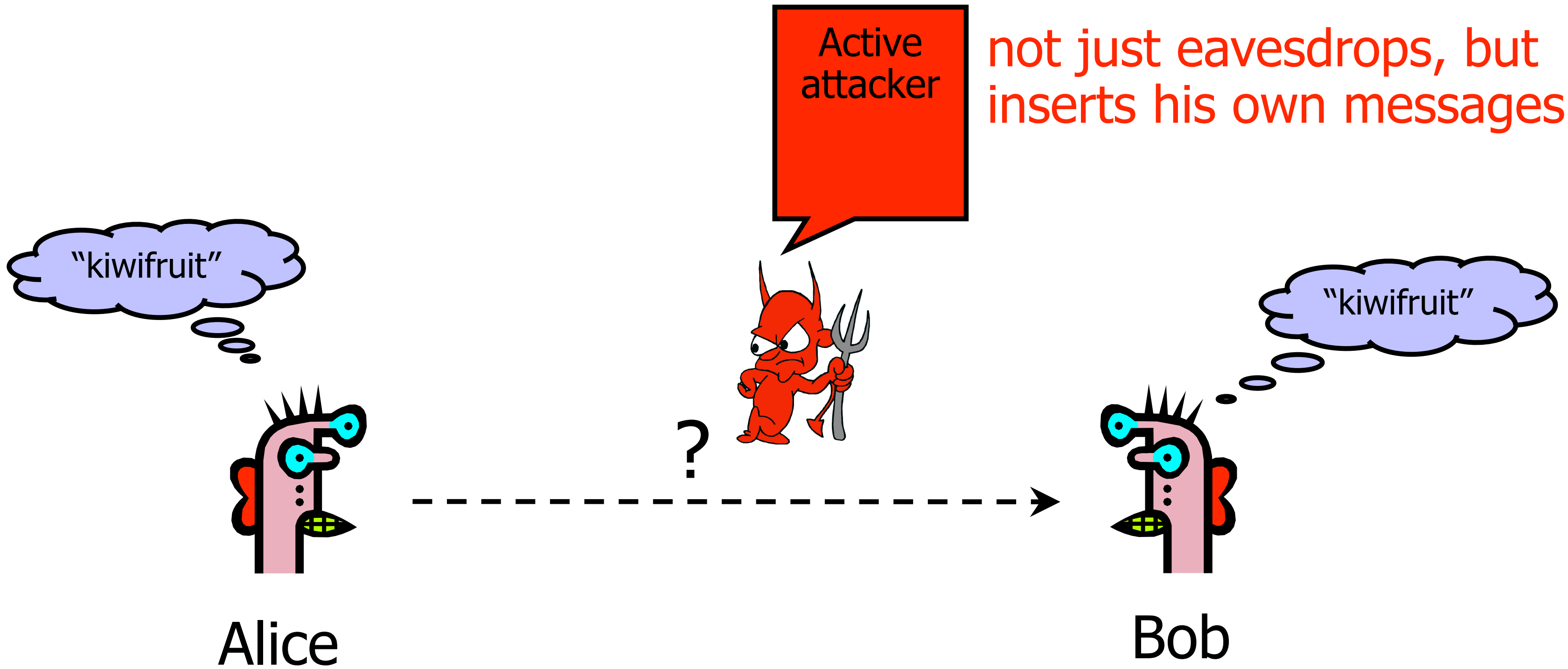
MIG-in-the-Middle Attack [Ross Anderson]



MIG-in-the-Middle Attack [Ross Anderson]



Authentication with Shared Secret

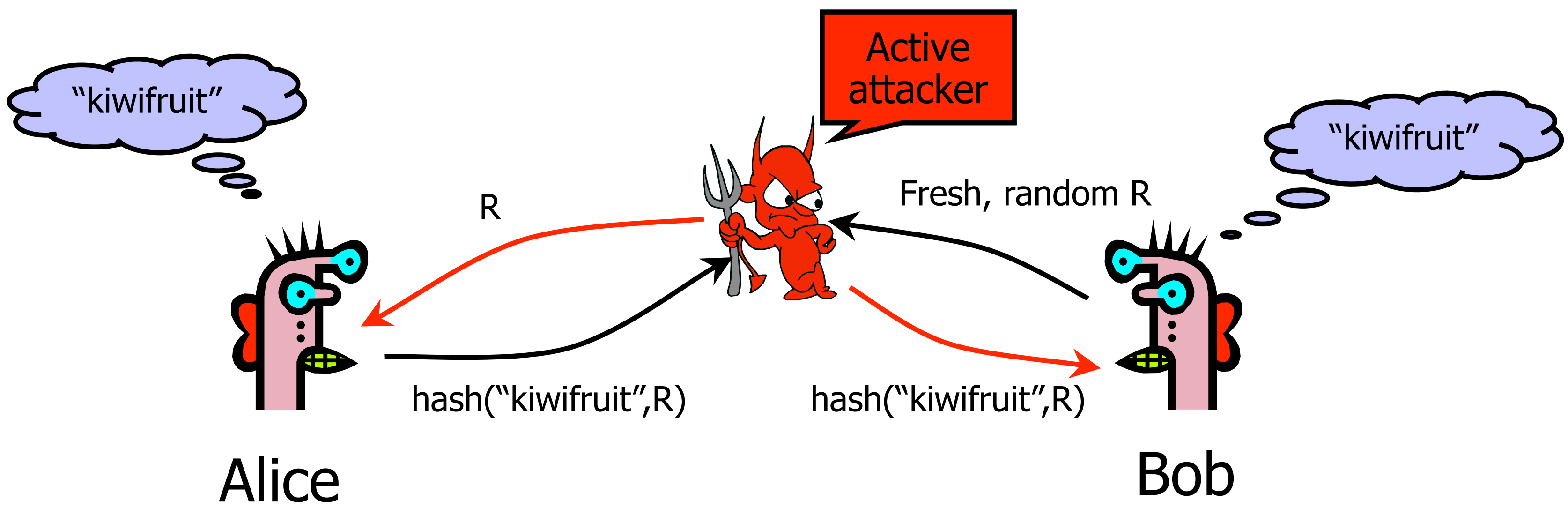


Alice and Bob share some secret.

How can they identify each other on the network?

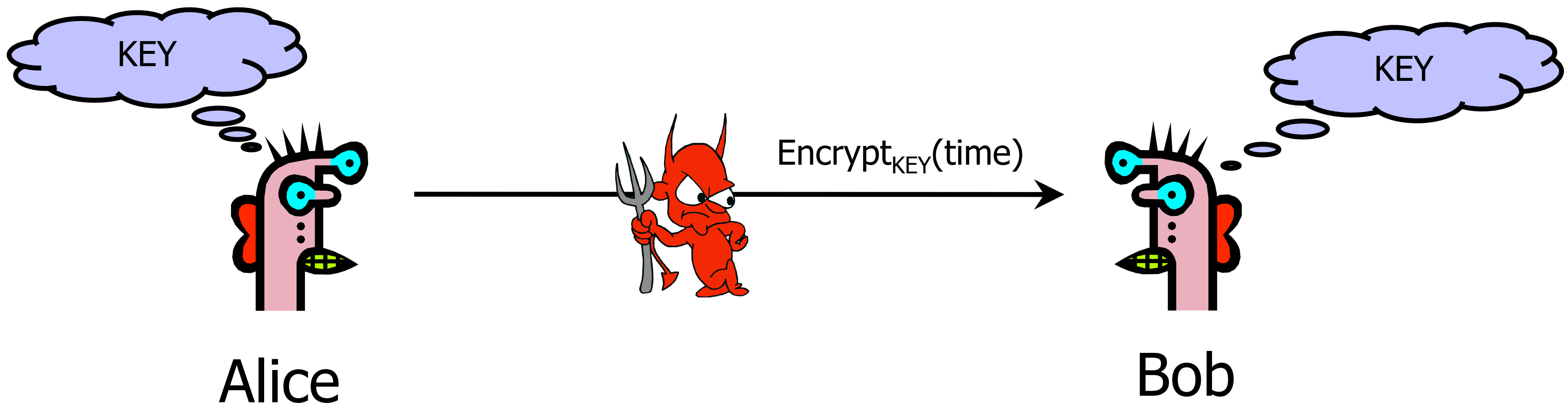
What have we learned from the systems we've seen?

Challenge-Response

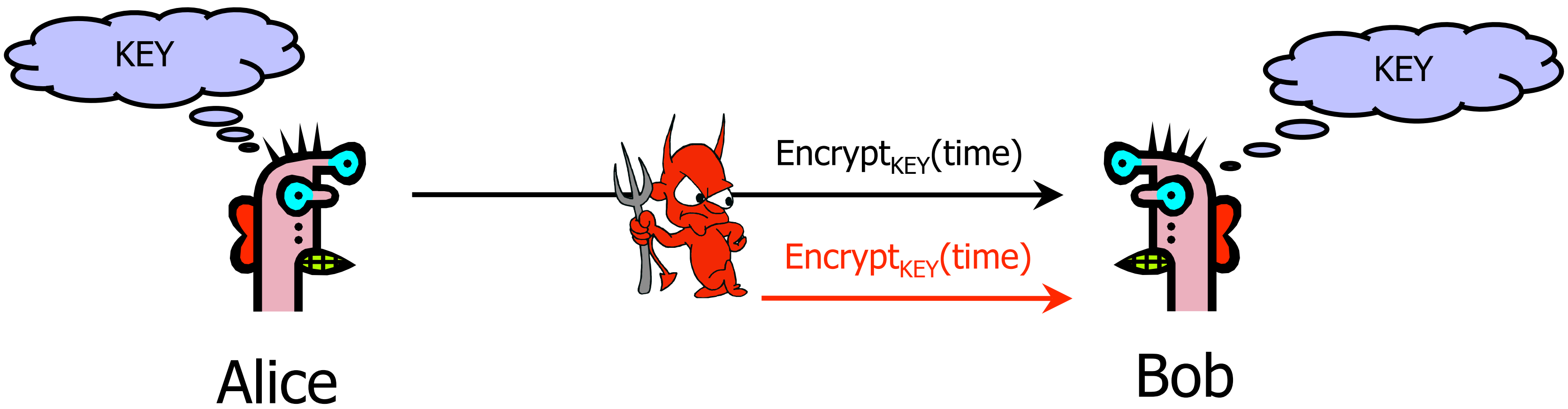


- Man-in-the-middle attack on challenge-response
  - Attacker successfully authenticates as Alice by simple replay
- This is an attack on authentication, not secrecy
  - Attacker does <u>not</u> learn the shared secret
- However, response opens the door to offline dictionary attack

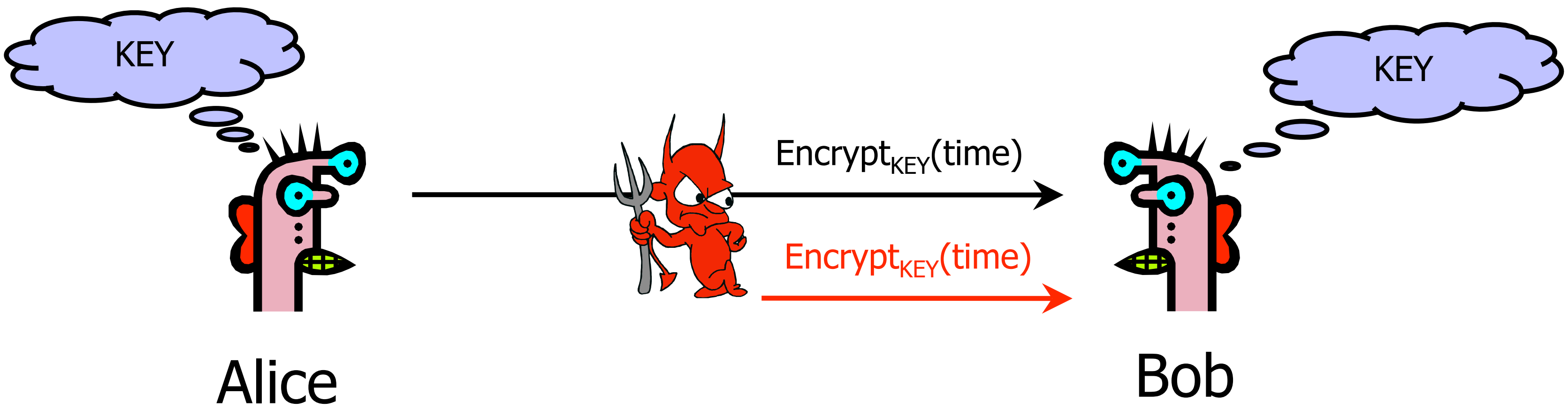
Encrypted Timestamp



Encrypted Timestamp

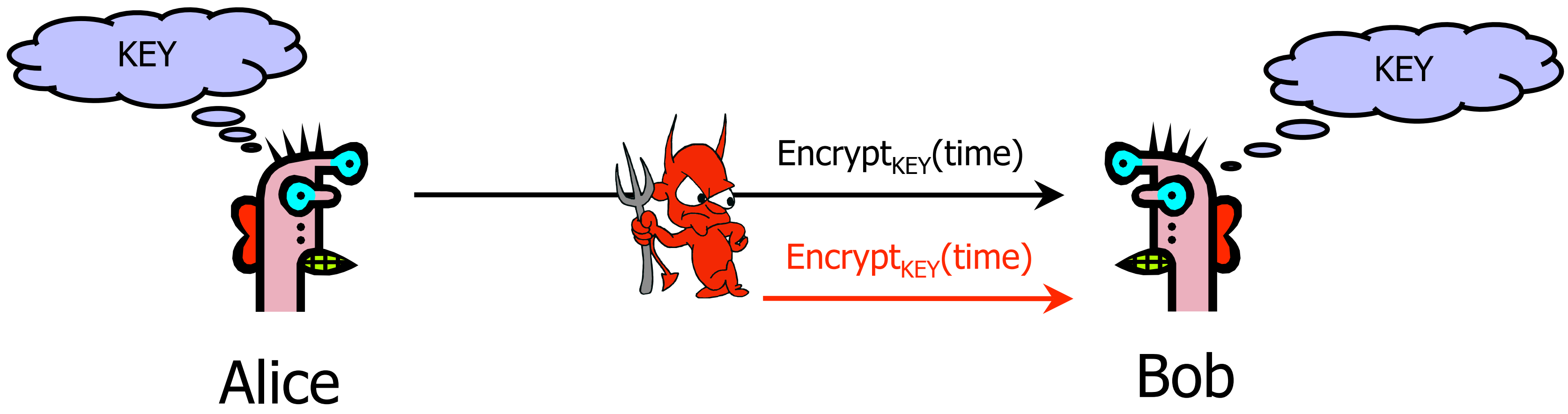


Encrypted Timestamp



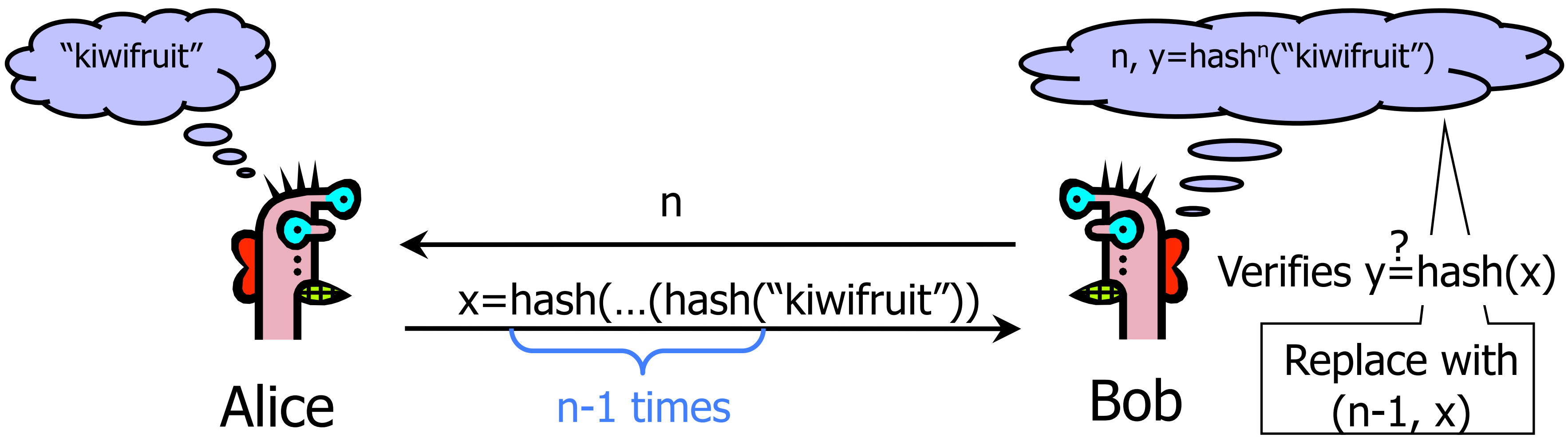
- Requires synchronized clocks
  - Bob's clock must be secure, or else attacker will roll it back and reuse an old authentication message from Alice

Encrypted Timestamp



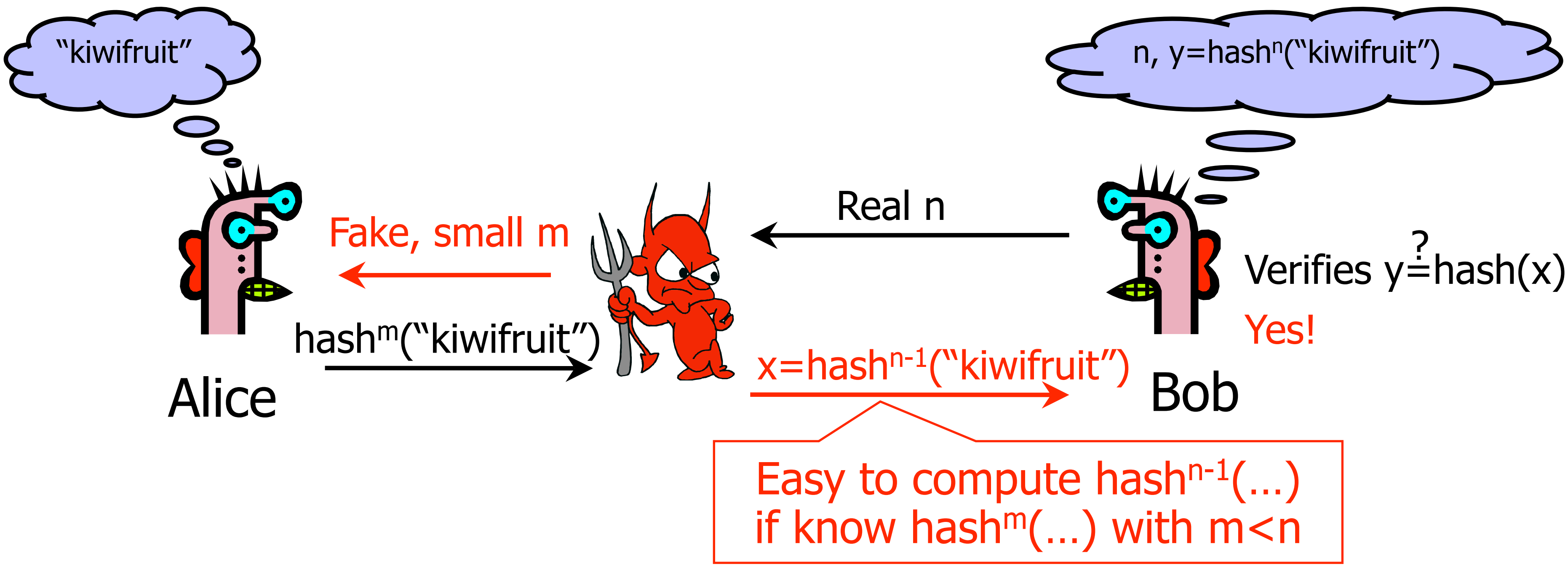
- Requires synchronized clocks
  - Bob's clock must be secure, or else attacker will roll it back and reuse an old authentication message from Alice
- Attacker can replay within clock skew window

Lamport's Hash



- Main idea: "hash stalk"
  - Moving up the stalk (computing the next hash) is easy, moving down the stalk (inverting the hash) is hard
  - n should be large (can only use it for n authentications)
- For verification, only need the tip of the stalk

"Small n" Attack



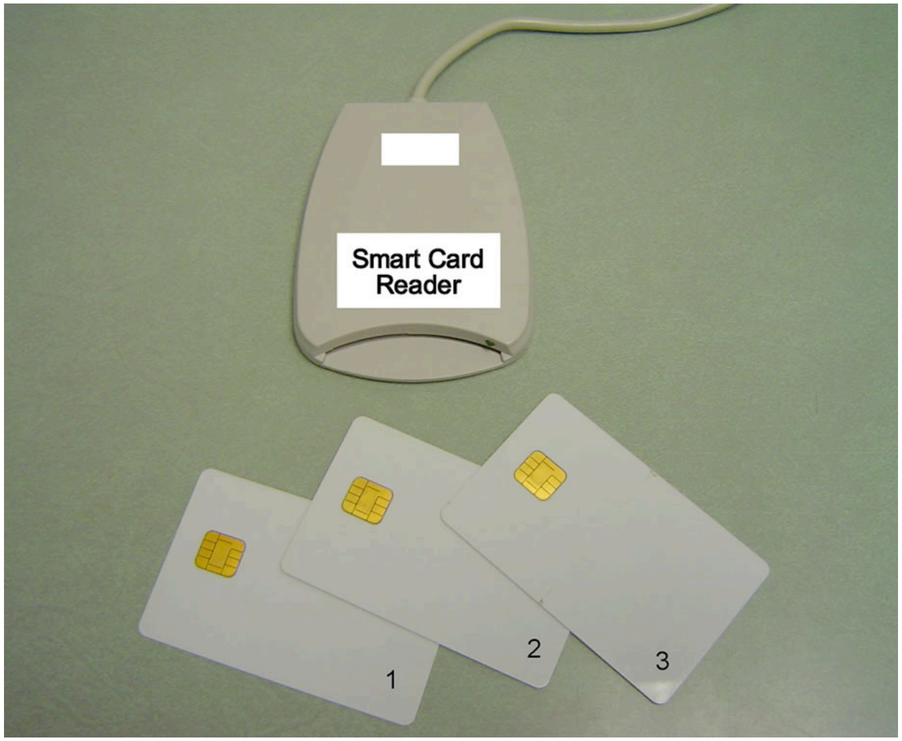
- First message from Bob is not authenticated!
- Alice should remember current value of n

Adversaries To Consider

- Eavesdropper
- Pretend to be Bob and accept connections from Alice
- Initiate conversation pretending to be Alice
- Read Alice's database
- Read Bob's database
- Modify messages in transit between Alice and Bob
- Any combination of the above

What You Have

- Smartcard
  - Little computer chip in credit card form factor



Smartcard Bank Cards [Drimer and Murdoch]

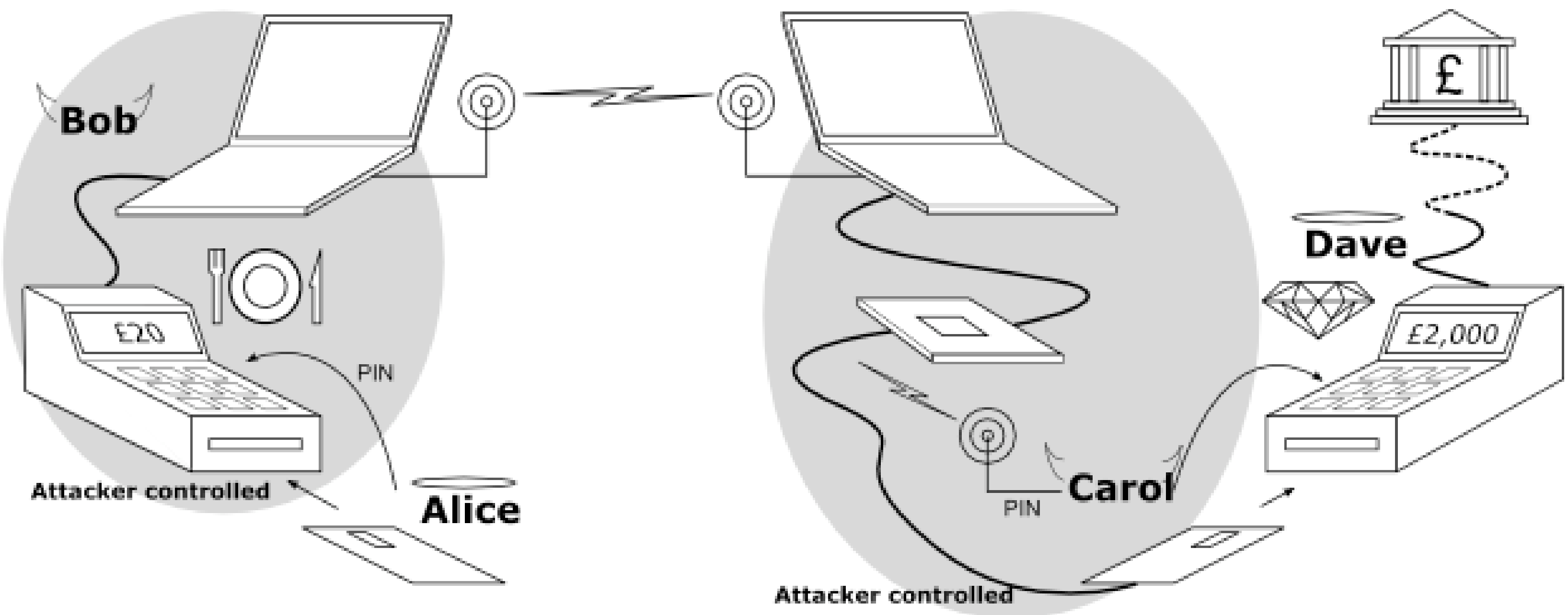


Image from http://www.cl.cam.ac.uk/research/security/projects/banking/relay/

Smartcard Bank Cards [Drimer and Murdoch]

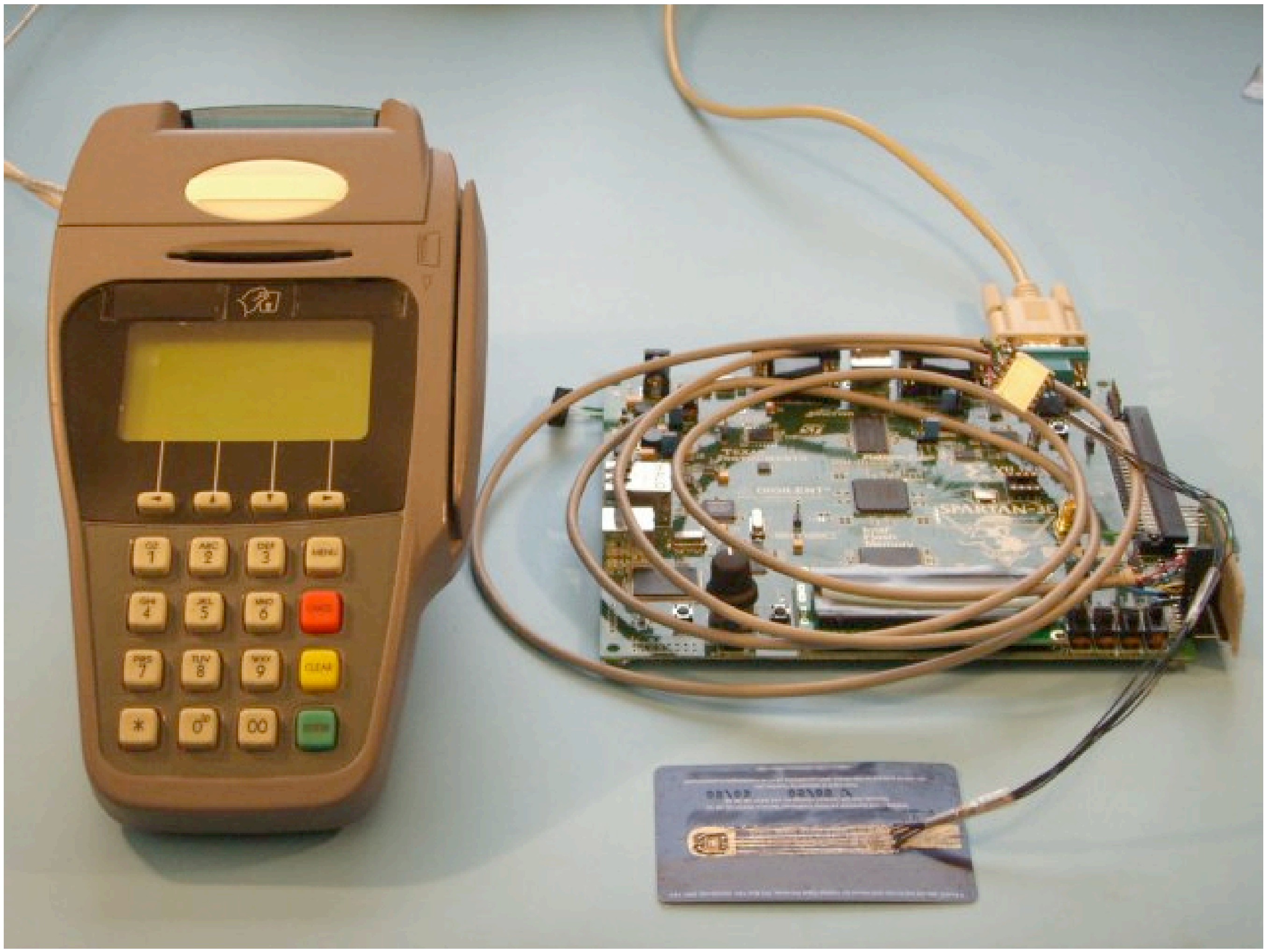


Image from http://www.cl.cam.ac.uk/research/security/projects/banking/relay/

Magstripe Writer

http://www.tyner.com/magnetic/msr206-1.jpg