CSE 484 (Winter 2010)

Software Security: Attacks, Defenses, and Design Principles

Tadayoshi Kohno

Thanks to Dan Boneh, Dieter Gollmann, John Manferdelli, John Mitchell, Vitaly Shmatikov, Bennet Yee, and many others for sample slides and materials ...

Goals for Today

- Defensive Approaches
- Cryptography Overview

Genetic Diversity

- Problems with Monoculture

- Steps toward diversity
  - Automatic diversification of compiled code
  - Address Space Randomization

Principles

- Check inputs

Principles

- Least privilege

Principles

- Check all return values

Principles

- Securely clear memory (passwords, keys, etc)

Principles

- Failsafe defaults

Principles

- Reduce size of TCB
- Simplicity
- Modularity

Principles

- Open design? Open source?
- Maybe...

- Linux Kernel Backdoor Attempt: http://www.freedom-to-tinker.com/?p=472

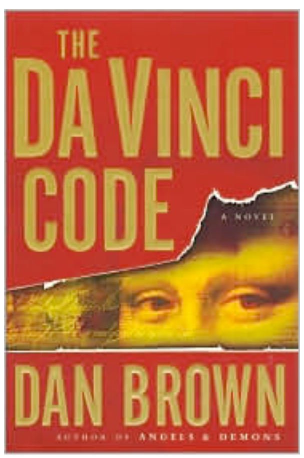
- PGP Corporation: http://www.pgp.com/developers/sourcecode/index.html

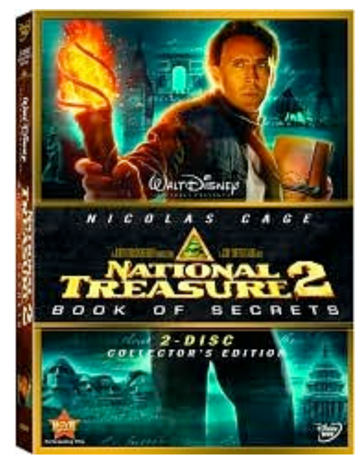
Vulnerability Analysis and Disclosure

- What do you do if you've found a security problem in a real system?
- Say
  - Electronic voting machine?
  - Boeing 787?
  - iPhone?
  - School grade database?

Cryptography and Security

- Art and science of *protecting* our *information*.
  - Keeping it private, if we want privacy
  - Protecting its integrity, if we want to avoid forgeries.



Images from Wikipedia and Barnes and Noble

Some thoughts about cryptography

- Cryptography only one small piece of a larger system
- Must protect entire system
  - Physical security
  - Operating system security
  - Network security
  - Users
  - **Cryptography** (following slides)
- "Security only as strong as the weakest link"
  - Need to secure weak links
  - But not always clear what the weakest link is (different adversaries and resources, different adversarial goals)
  - Crypto failures may not be (immediately) detected
- Cryptography helps after you've identified your threat model and goals

Improved se

- RFIDs in car ke
  - RFIDs in car k
  - Result: Car ja

boingboing

A Directory Of Wonderful Things

Biometric car lock defeated by cutting off owner's finger

POSTED BY CORY DOCTOROW, MARCH 31, 2005 7:53 AM | PERMALINK

Andrei sez, "'Malaysia car thieves steal finger.' This is what security visionaries Bruce Schneier and Ross Anderson have been warning about for a long time. Protect your $75,000 Mercedes with biometrics and you risk losing whatever body part is required by the biometric mechanism."

> ...[H]aving stripped the car, the thieves became frustrated when they wanted to restart it. They found they again could not bypass the immobiliser, which needs the owner's fingerprint to disarm it.
>
> They stripped Mr Kumaran naked and left him by the side of the road - but not before cutting off the end of his index finger with a machete.

Key Entry Pad (4-digit PIN)

- This is the key pad on my office safe.
- Inside my safe is a copy of final exam.
- How long would it take a you to break in?

- Answer (combinatorics):
  - 10^4 tries *maximum*.
  - 10^4 / 2 tries on *average*.
- Answer (unit conversion):
  - 3 seconds per try --> 4 hours and 10 minutes on average

Image from profmason.com

Key Entry Pad (4-digit PIN)

Image from profmason.com

- Now assume the safe automatically calls police after 3 failed attempts.
- What is the probability that you will guess the PIN within 3 tries?
- (Assume no repeat tries.)

- Answer (combinatorics):
  - 10000 choose 3 possible choices for the 3 guesses
  - $1 \times \left(9999 \text{ choose } 2\right)$ possible choices contain the correct PIN
  - So success probability is 3 / 10000

Key Entry Pad (4-digit PIN)

- Could you do better at guessing the PIN?

- Answer (*chemical* combinatorics):
  - Put different chemical on each key (NaCl, KCl, LiCl, ...)

Image from profmason.com

Idea from http://eprint.iacr.org/2003/217.ps

Key Entry Pad (4-digit PIN)

- Couldyou do better at guessing the PIN?

- Answer (*chemical* combinatorics):
  - Put different chemical on each key (NaCl, KCl, LiCl, ...)
  - Observe residual patterns after I access safe

Image from profmason.com

Idea from http://eprint.iacr.org/2003/217.ps

Key Entry Pad (4-digit PIN)

- Could you do better at guessing the PIN?

- Answer (*chemical* combinatorics):
  - Put different chemical on each key (NaCl, KCl, LiCl, ...)
  - Observe residual patterns after I access safe

Image from profmason.com

Idea from http://eprint.iacr.org/2003/217.ps

Key Entry Pad (4-digit PIN)

- Could you do better at guessing the PIN?
  - Answer (*chemical* combinatorics):
    - Put different chemical on each key (NaCl, KCl, LiCl, ...)
    - Observe residual patterns after I access safe

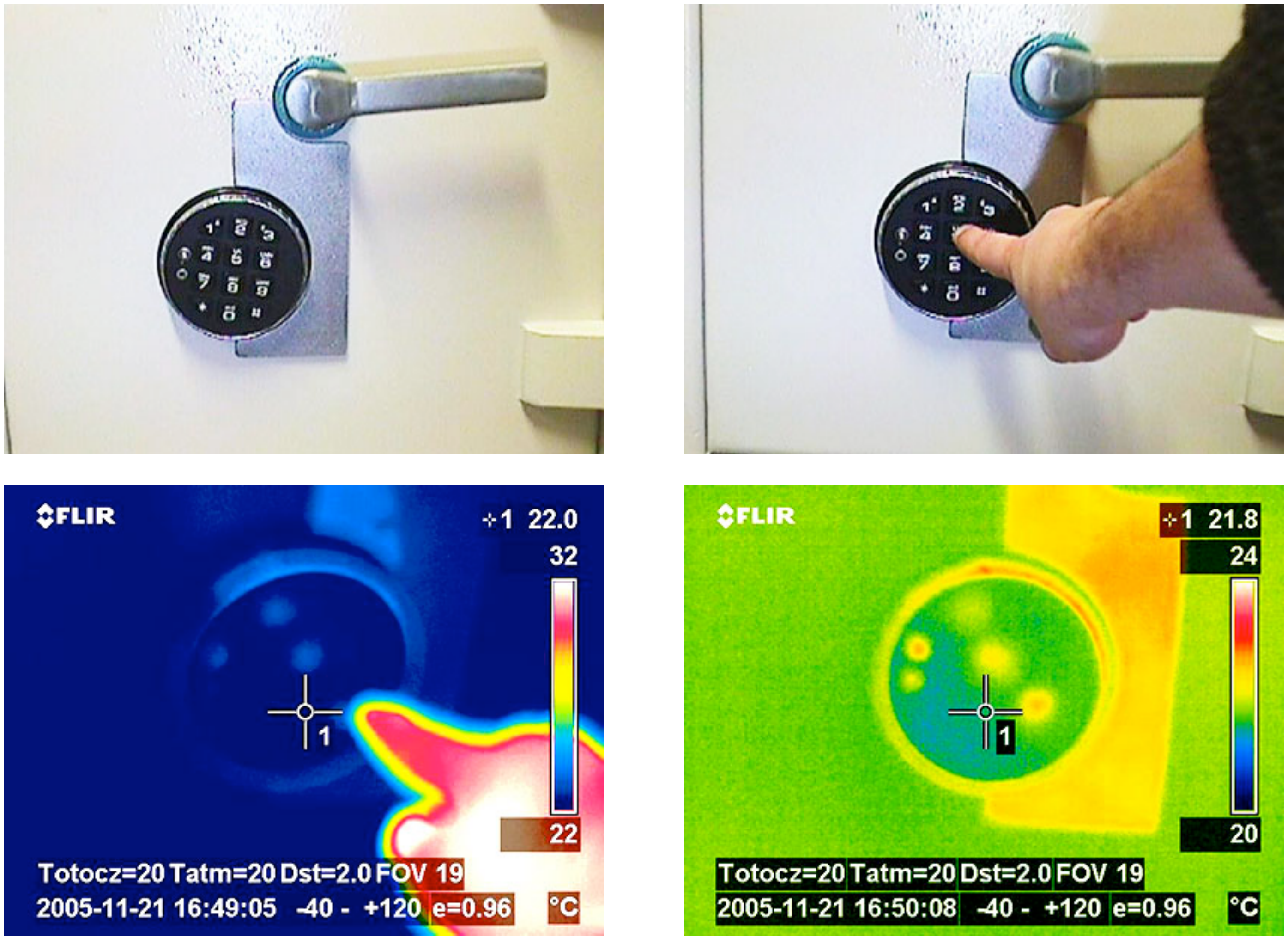
Lesson: Consider the complete system, physical security, etc

Lesson: Think outside the box

Image from profmason.com

Idea from http://eprint.iacr.org/2003/217.ps

Thermal Patterns

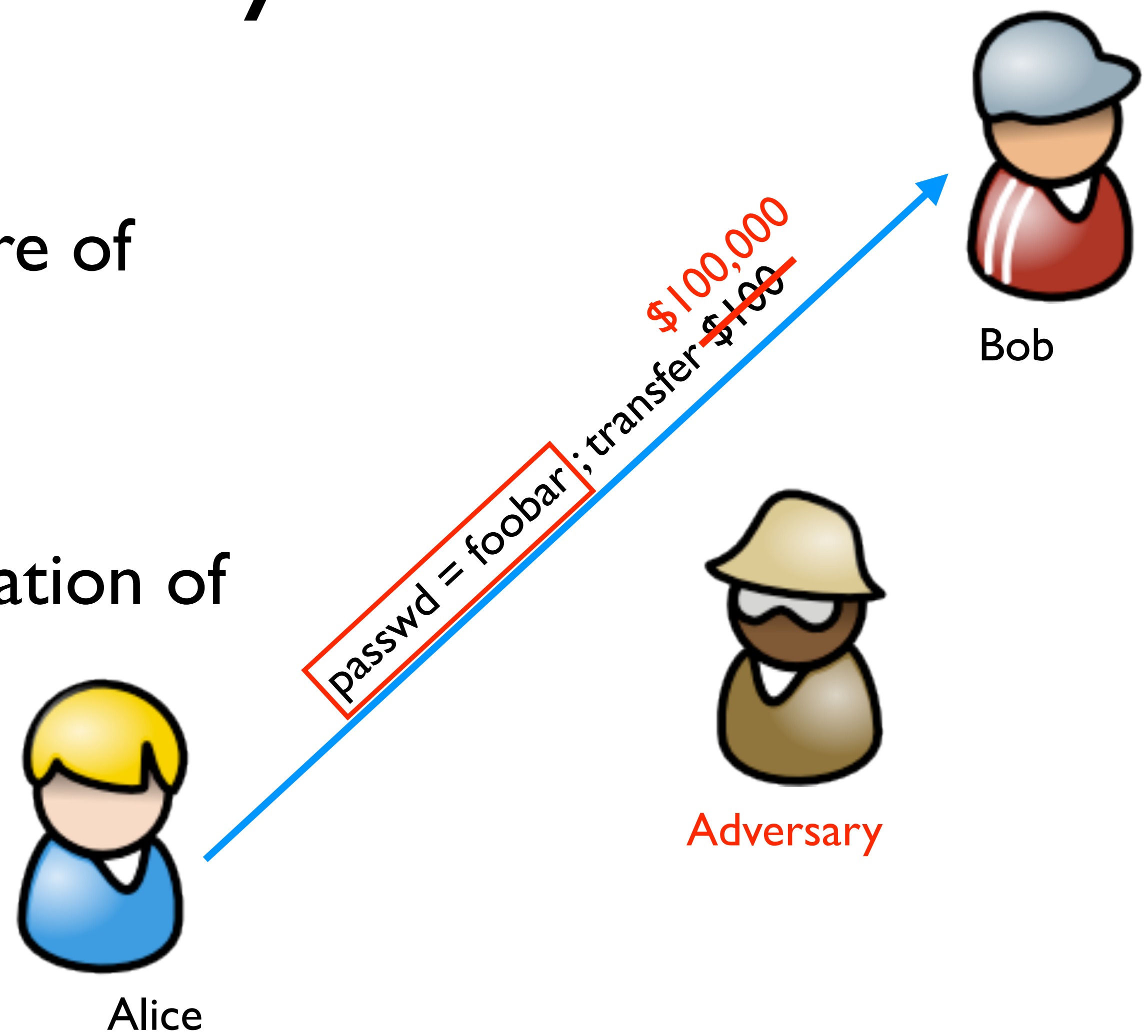


Images from http://lcamtuf.coredump.cx/tsafe/

Common Communication Security Goals

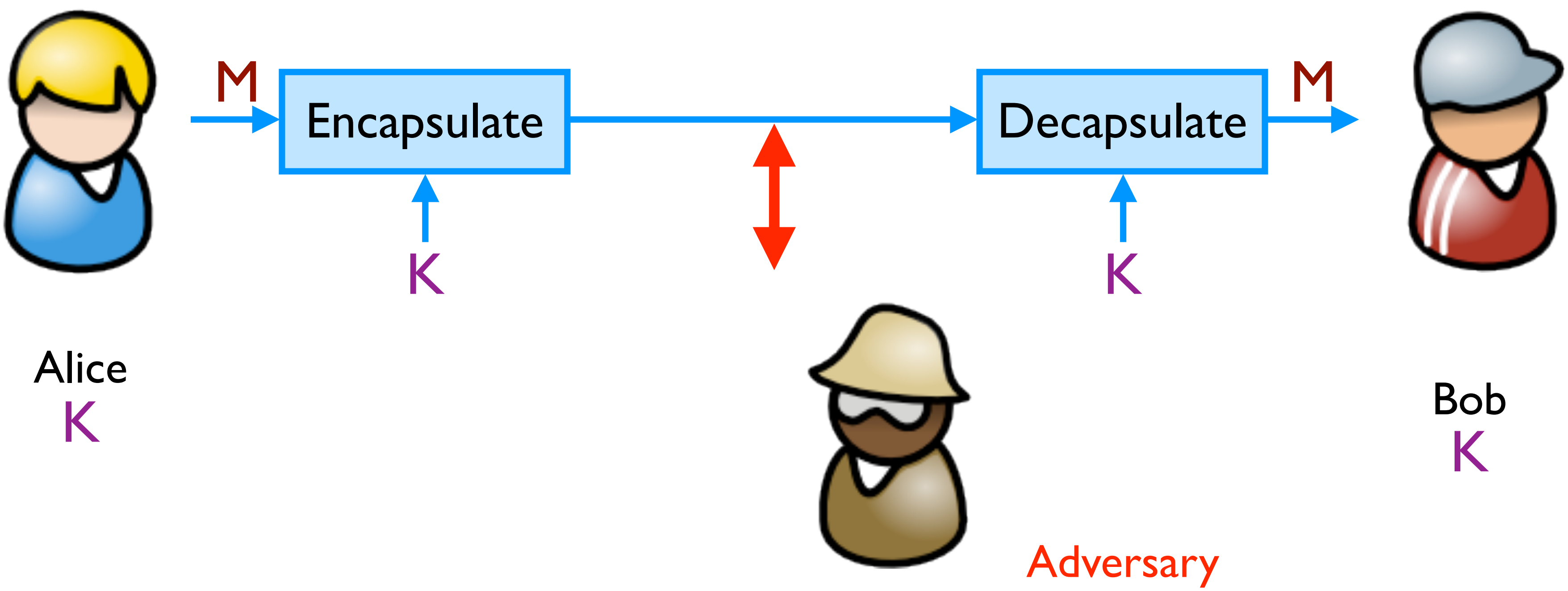
Privacy of data
Prevent exposure of information

Integrity of data
Prevent modification of information



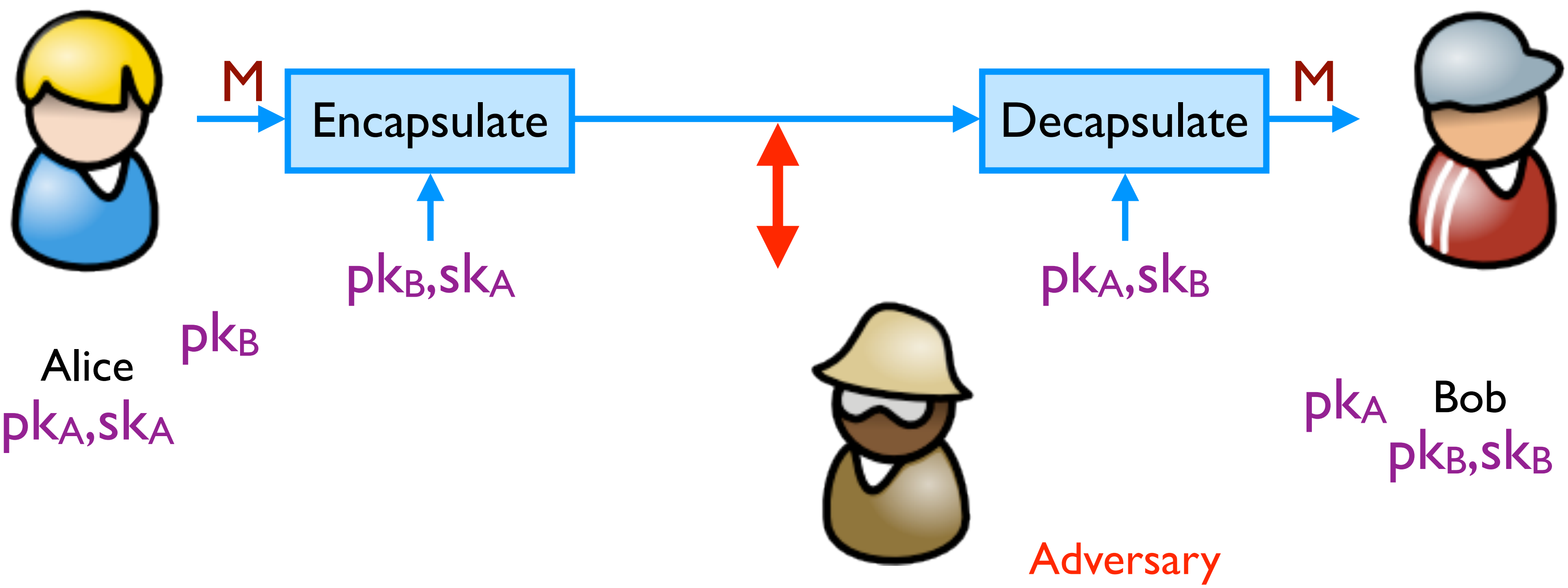
Symmetric Setting

Both communicating parties have access to a shared random string K, called the key.



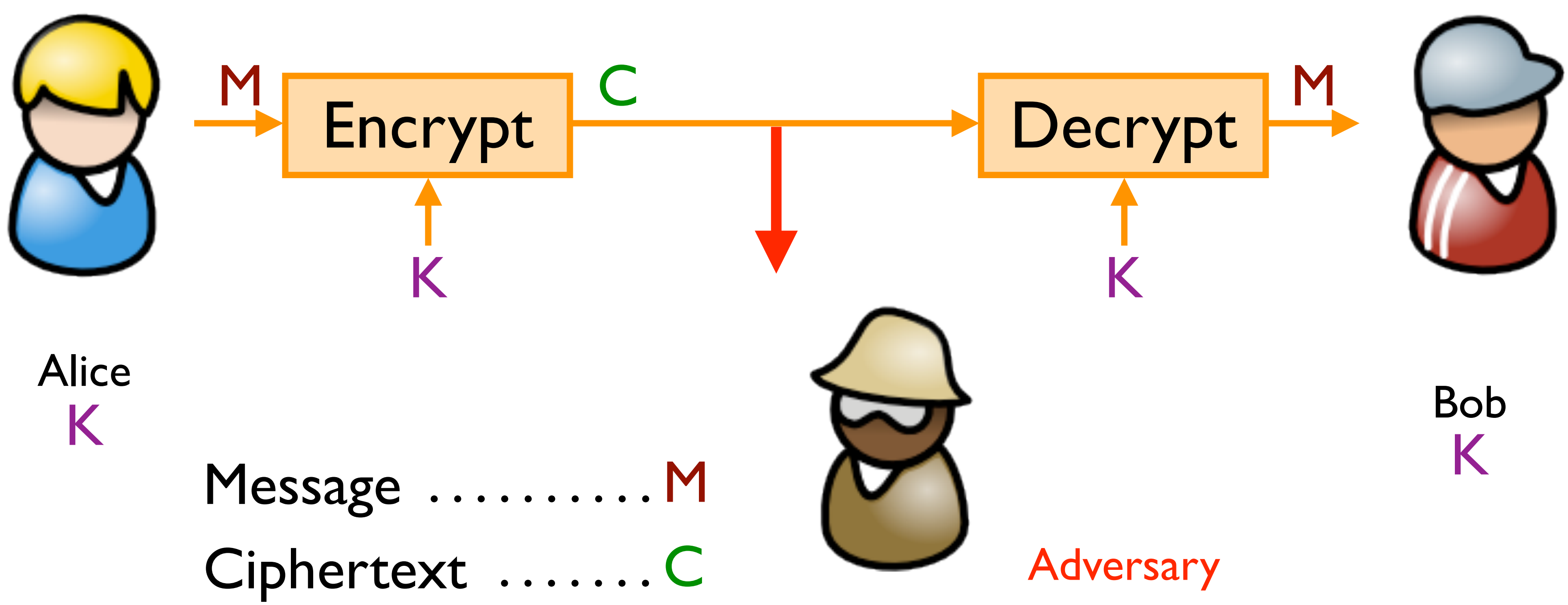
Asymmetric Setting

Each party creates a public key pk and a secret key sk.



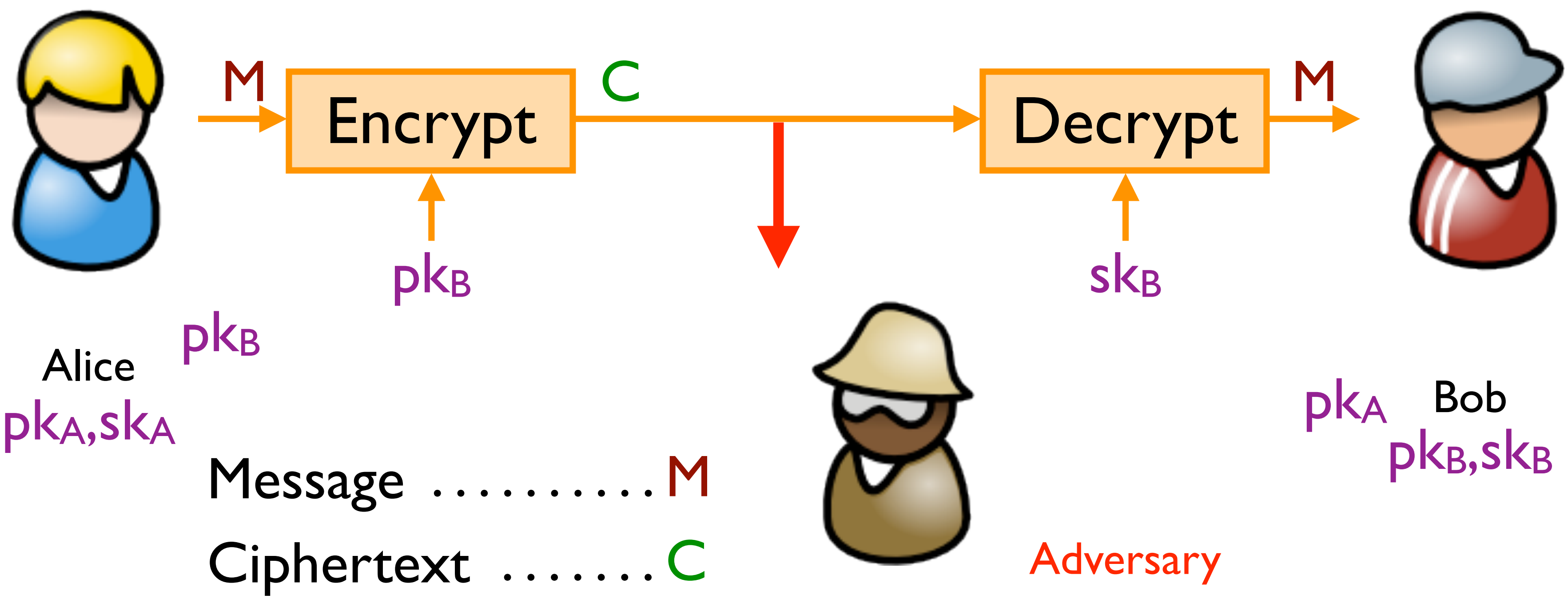
Achieving Privacy (Symmetric)

Encryption schemes: A tool for protecting privacy.



Achieving Privacy (Asymmetric)

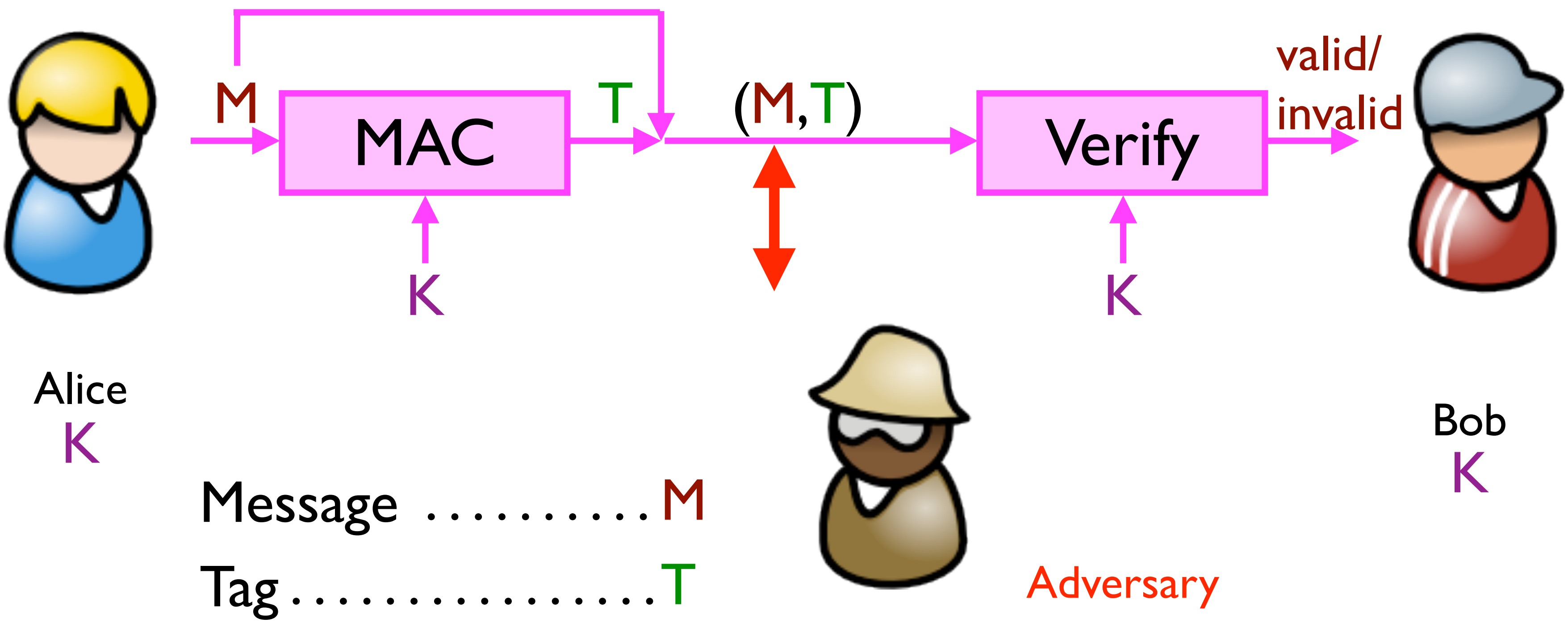
Encryption schemes: A tool for protecting privacy.



Achieving Integrity (Symmetric)

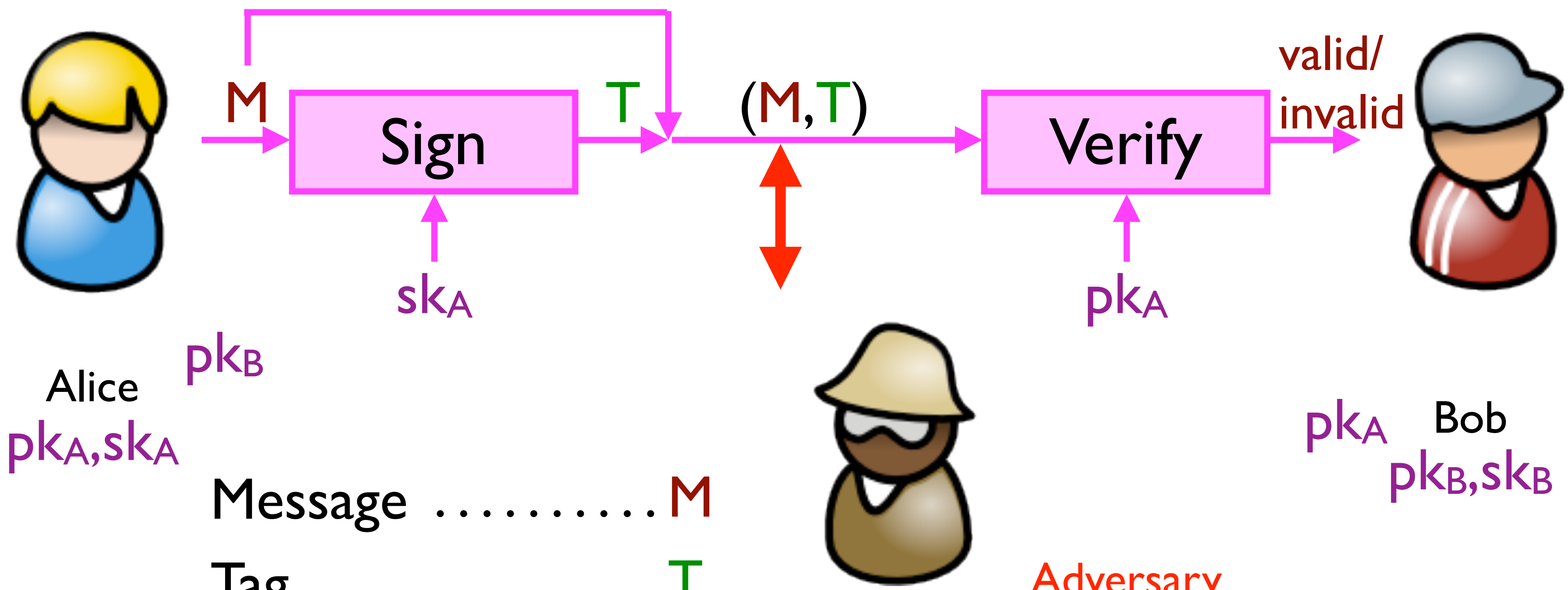
Message authentication schemes: A tool for protecting integrity.

(Also called message authentication codes or MACs.)



Achieving Integrity (Asymmetric)

Digital signature schemes: A tool for protecting integrity and authenticity.

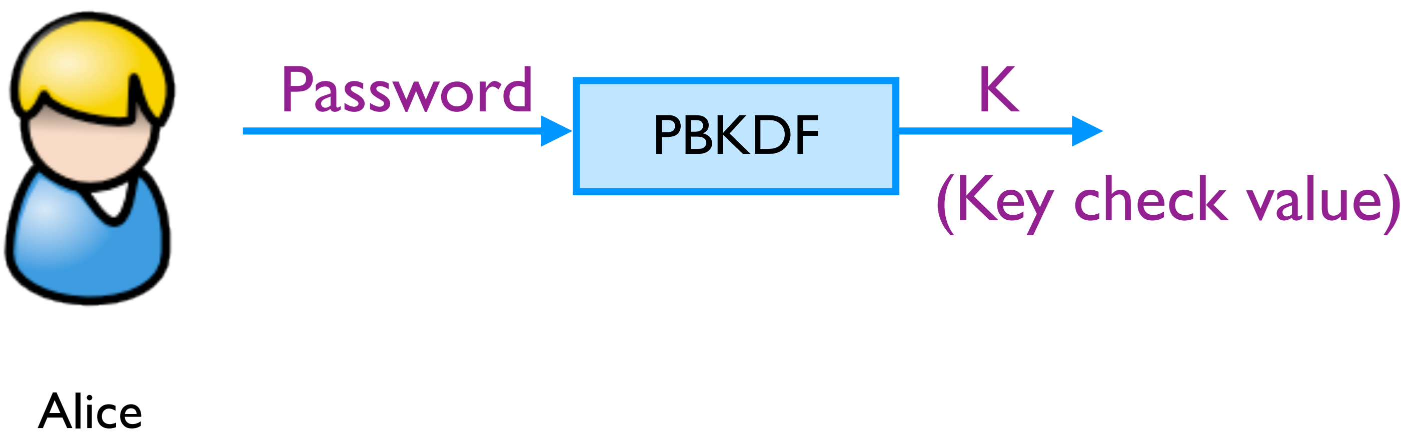


Message . . . . M

Tag . . . . T

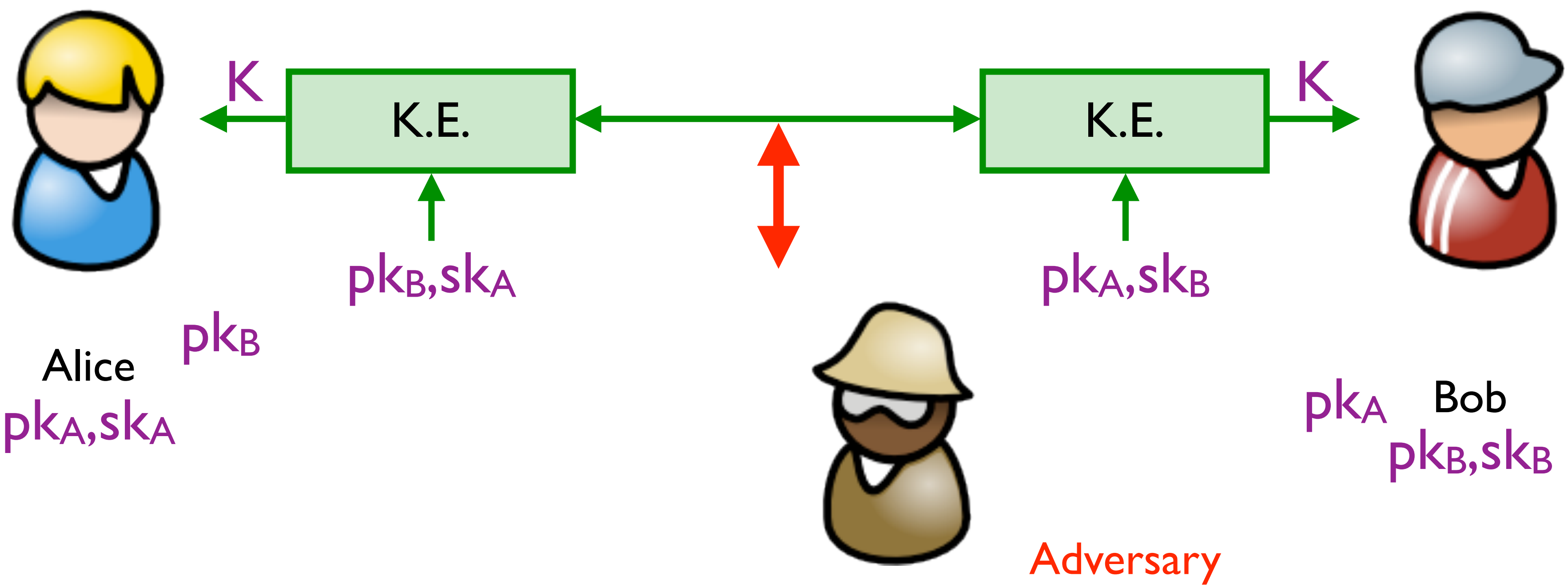
Getting keys: PBKDF

Password-based Key Derivation Functions



Getting keys: Key exchange

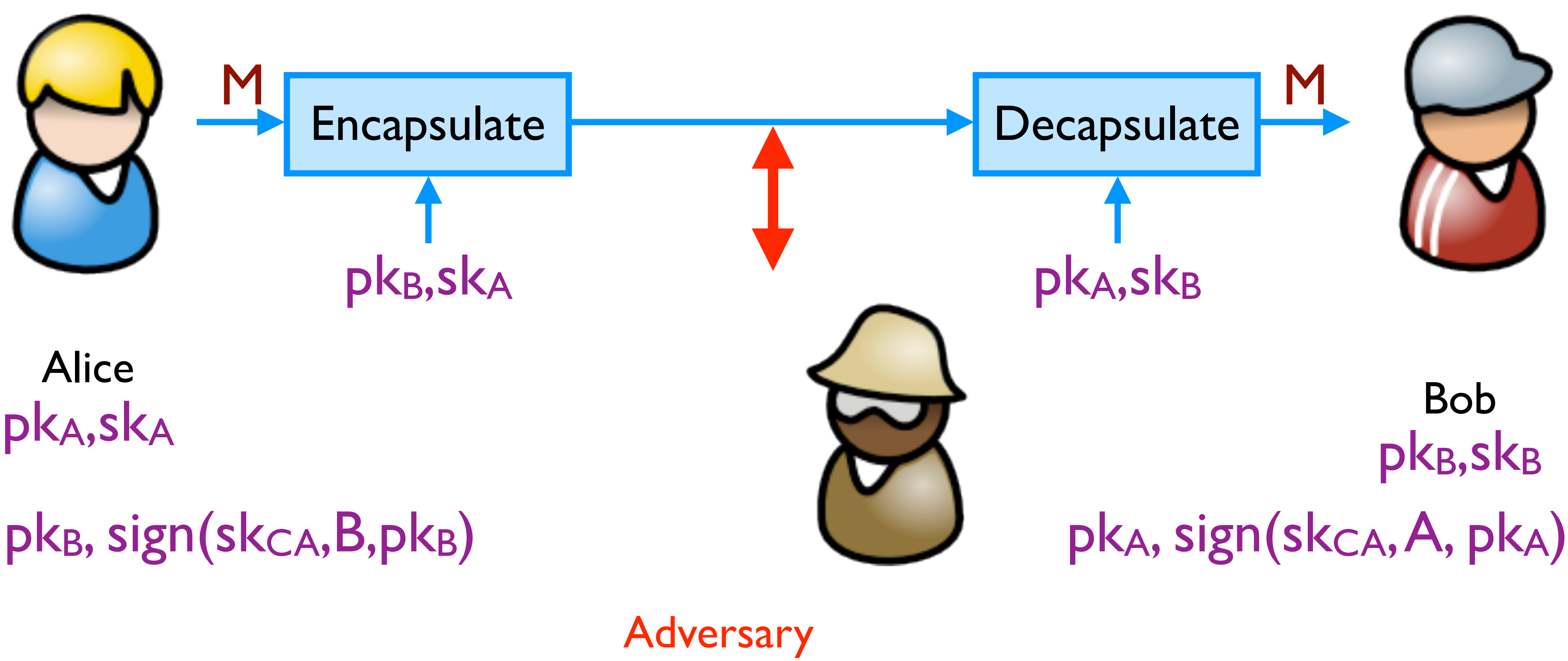
Key exchange protocols: A tool for establishing a share symmetric key



Getting keys: CAs

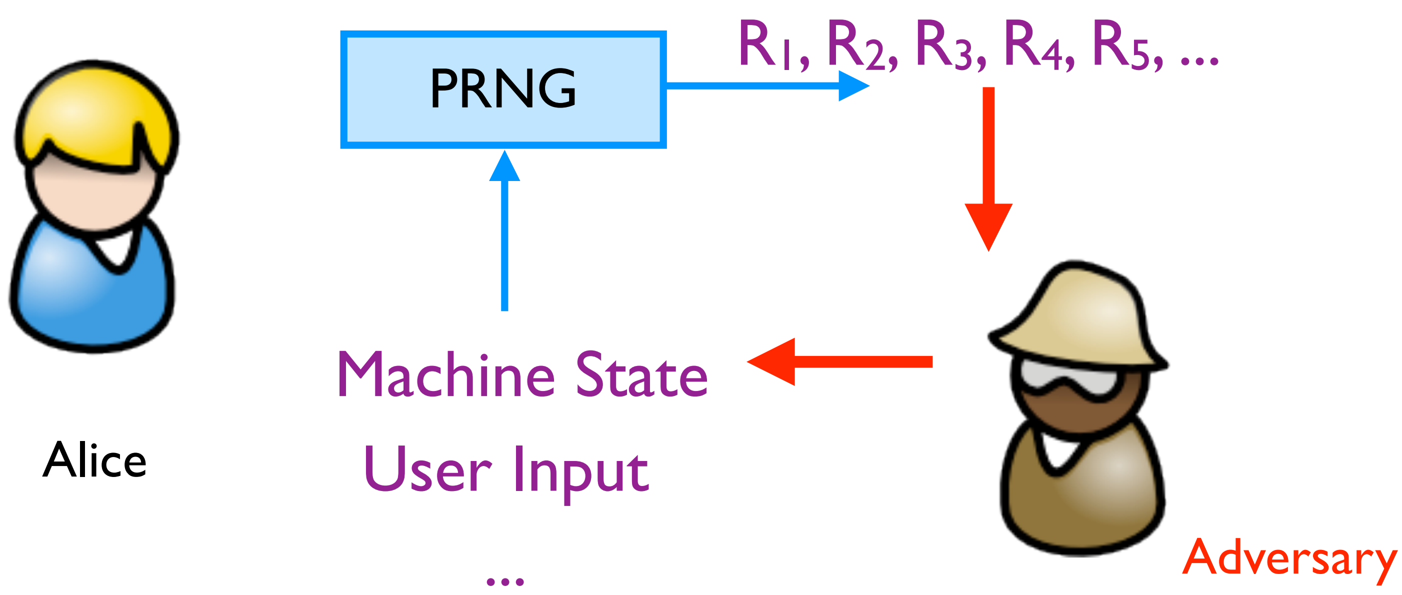
Each party creates a public key pk and a secret key sk.

(Public keys signed by a trusted third party: a certificate authority.)

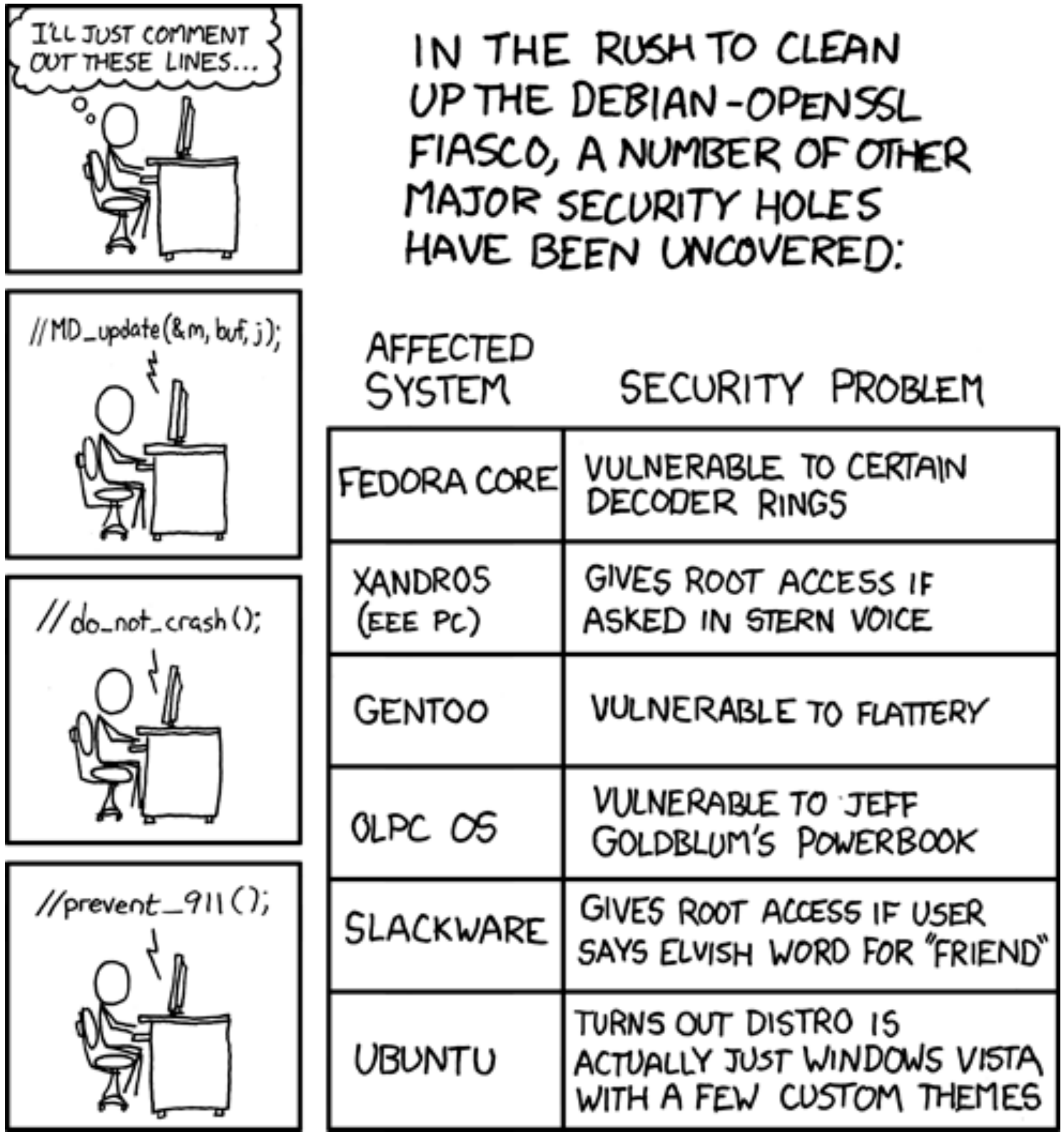


“Random” Numbers

Pseudorandom Number Generators (PRNGs)




DILBERT By Scott Adams
TOUR OF ACCOUNTING
OVER HERE WE HAVE OUR RANDOM NUMBER GENERATOR.
scottadams@aol.com
www.dilbert.com
NINE NINE NINE NINE NINE NINE
© 2001 United Feature Syndicate, Inc.
ARE YOU SURE THAT'S RANDOM?
THAT'S THE PROBLEM WITH RANDOMNESS: YOU CAN NEVER BE SURE.


I'LL JUST COMMENT OUT THESE LINES...
//MD_update(&m, buf, j);
//do_not_crash();
//prevent_911();
IN THE RUSH TO CLEAN UP THE DEBIAN-OPENSSL FIASCO, A NUMBER OF OTHER MAJOR SECURITY HOLES HAVE BEEN UNCOVERED:
AFFECTED SYSTEM
SECURITY PROBLEM
FEDORA CORE
VULNERABLE TO CERTAIN DECODER RINGS
XANDROS (EEE PC)
GIVES ROOT ACCESS IF ASKED IN STERN VOICE
GENTOO
VULNERABLE TO FLATTERY
OLPC OS
VULNERABLE TO JEFF GOLDBLUM'S POWERBOOK
SLACKWARE
GIVES ROOT ACCESS IF USER SAYS ELVISH WORD FOR "FRIEND"
UBUNTU
TURNS OUT DISTRO IS ACTUALLY JUST WINDOWS VISTA WITH A FEW CUSTOM THEMES

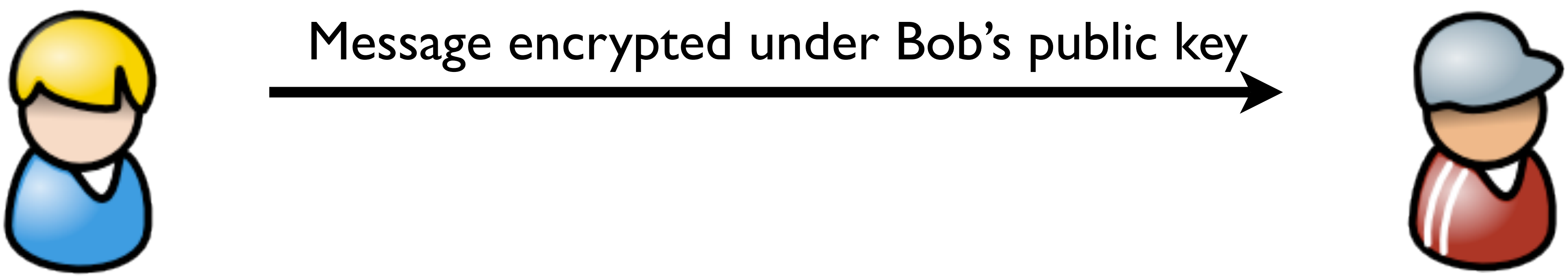
Source: XKCD

Kerckhoff's Principle

- Security of a cryptographic object should depend **only** on the secrecy of the secret (private) key
- Security should not depend on the secrecy of the algorithm itself.
- Why?

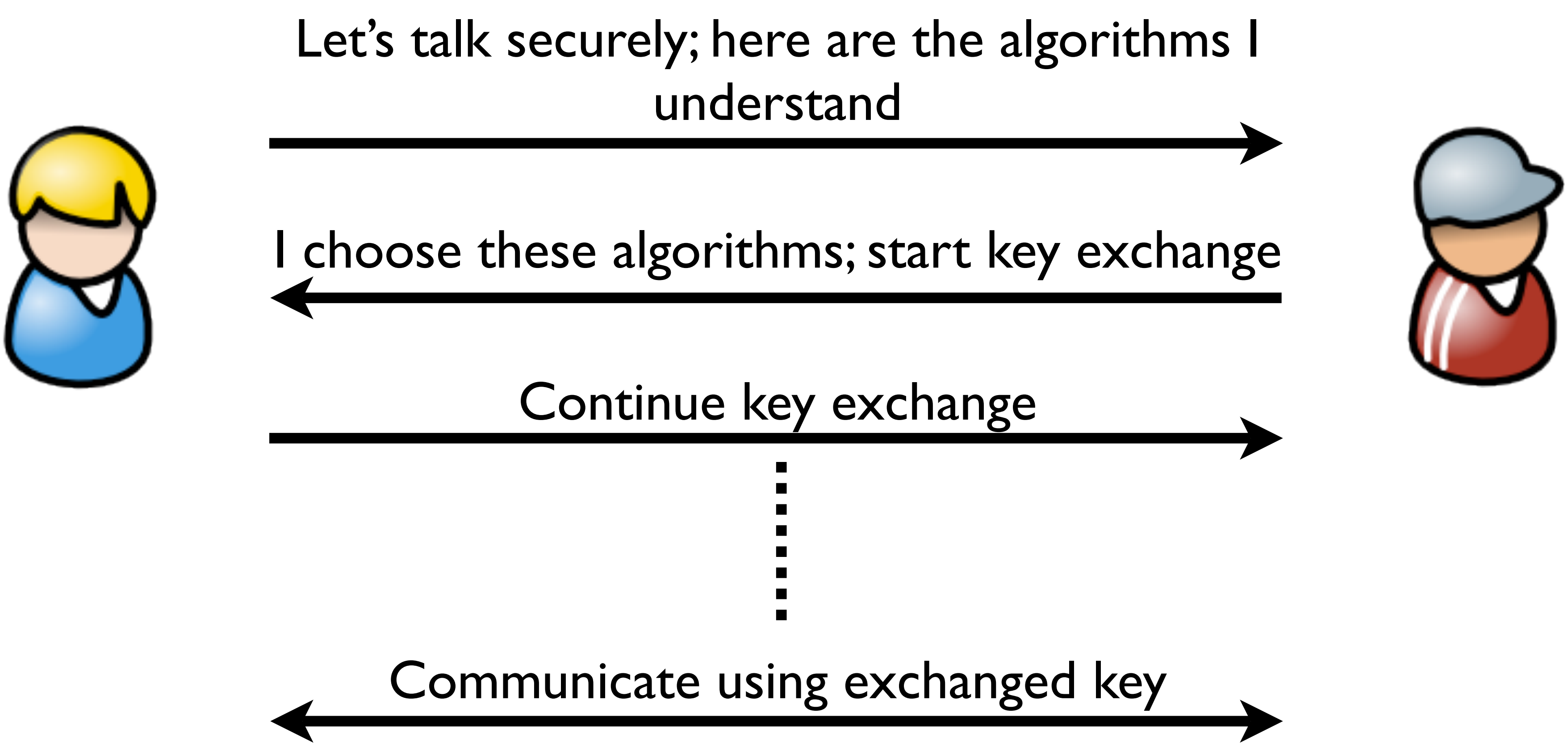
One-way Communications

PGP is a good example



Interactive Communications

In many cases, it's probably a good idea to just use a standard protocol/system like SSH, SSL/TLS, etc...



Let’s Dive a Bit Deeper

One-way Communications

(*Informal* example; ignoring, e.g., signatures)

1. Alice gets Bob's public key; Alice *verifies* Bob's public key (e.g., via CA)
2. Alice generates random symmetric keys K1 and K2
3. Alice encrypts the message M the key K1; call result C
4. Alice authenticates (MACs) C with key K2; call the result T
5. Alice encrypts K1 and K2 with Bob's public key; call the result D

6. Send D, C, T

(Assume Bob's private key is encrypted on Bob's disk.)

7. Bob takes his password to derive key K3
8. Bob decrypts his private key with key K3
9. Bob uses private key to decrypt K1 and K2
10. Bob uses K2 to verify MAC tag T
11. Bob uses K1 to decrypt C

Interactive Communications

(*Informal* example; details omitted)

1. Alice and Bob exchange public keys and certificates
2. Alice and Bob use CA's public keys to verify certificates and each other's public keys
3. Alice and Bob take their passwords and derive symmetric keys
4. Alice and Bob use those symmetric keys to decrypt and recover their asymmetric private keys.
5. Alice and Bob use their asymmetric private keys and a *key exchange* algorithm to derive a shared symmetric key

(They key exchange process will require Alice and Bob to generate new pseudorandom numbers)

6. Alice and Bob use shared symmetric key to encrypt and authenticate messages

(Last step will probably also use random numbers; will need to rekey regularly; may need to avoid replay attacks,...)