Practical Cryptographic Systems

Provable Security

Housekeeping

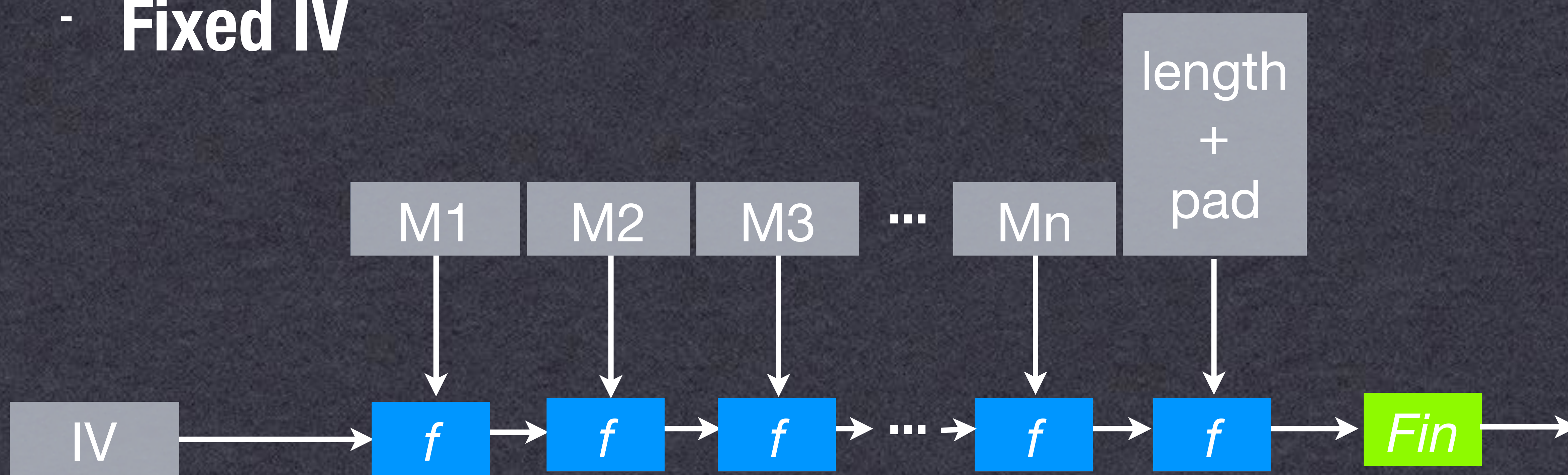
- Protocol and BAN Logic
  - DTCP Protocol
  - BAN Logic link is fixed
  - Midterm: 4/5
    - Open book, open notes

Review

- Last time:
  - Protocols (and how they fail)
  - Examples: SSL/TLS, DECT

Merkle-Damgård

- Used in most standard hash functions
  - (MDx, SHAx)
  - Fixed IV



f = compression function (m x n-bit input)

Fin = (optional) finalization function

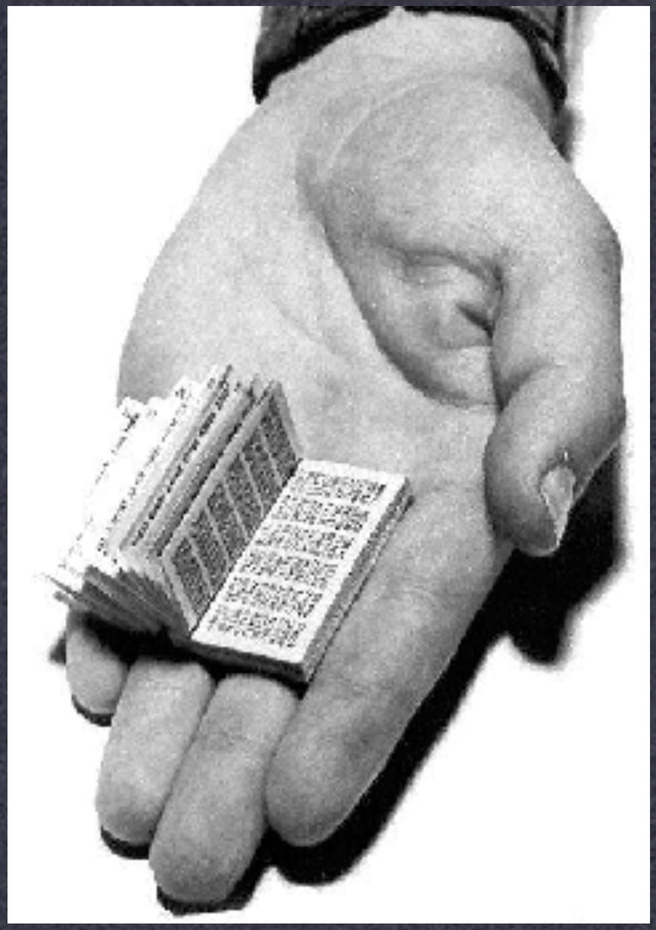
Today

- Provable security
  - How it informs system design
  - Why it’s used, why it’s (sometimes) ignored
  - How provable designs are often accidentally “broken”
  - More importantly: what does “provable” mean?

Information-Theoretic Security

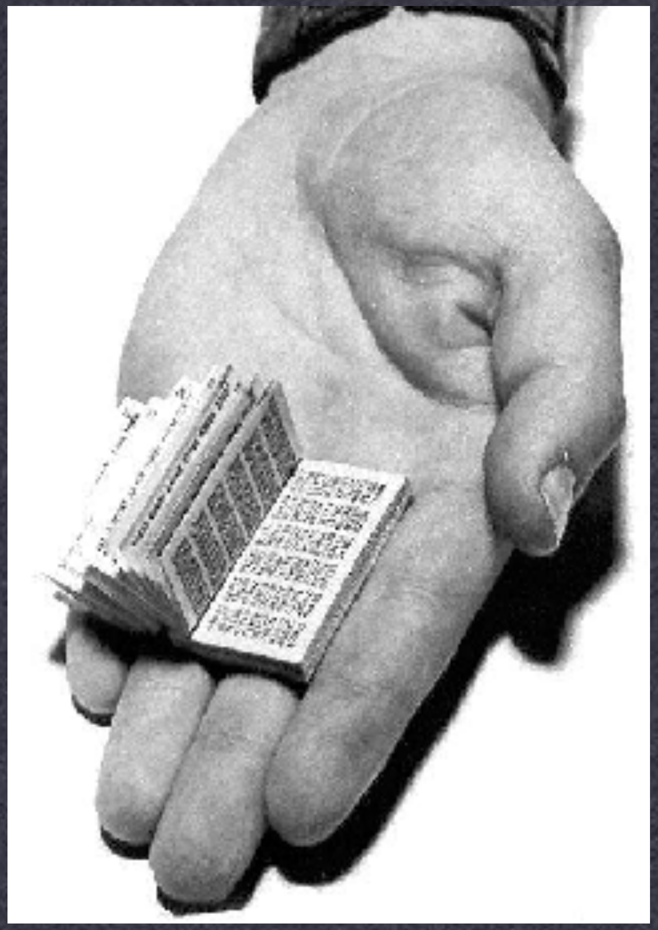
- Information Theoretic Security
  - (Vernam & Mauborgne OTP, Claude Shannon)
  - OTP Security Proof:

    Given a ciphertext, there is a
    <u>key</u> for each possible <u>plaintext</u>
    (and every key is equally likely!)



Information-Theoretic Security

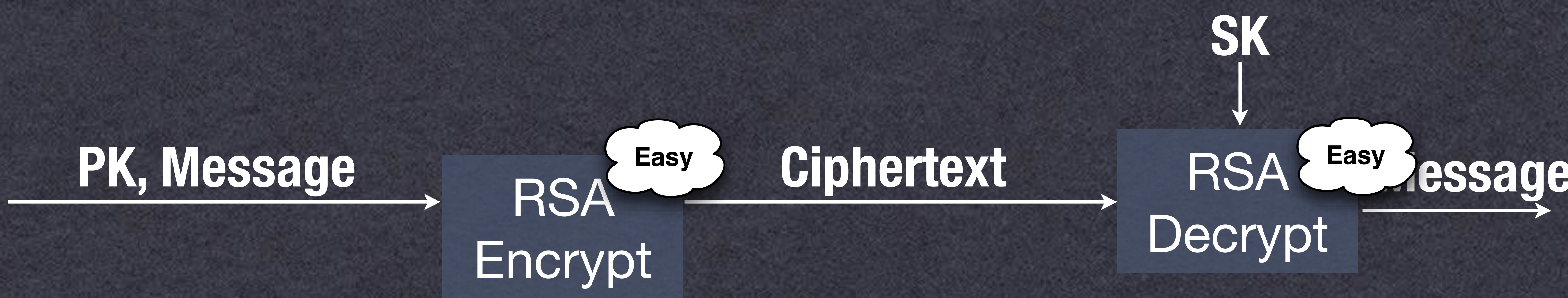
- Advantages:
  - Secure against <u>any amount</u> of effort
  - Brute-force attacks not possible
  - Requires no special assumptions (beyond correct implementation)
- Disadvantages
  - Most schemes pretty inefficient



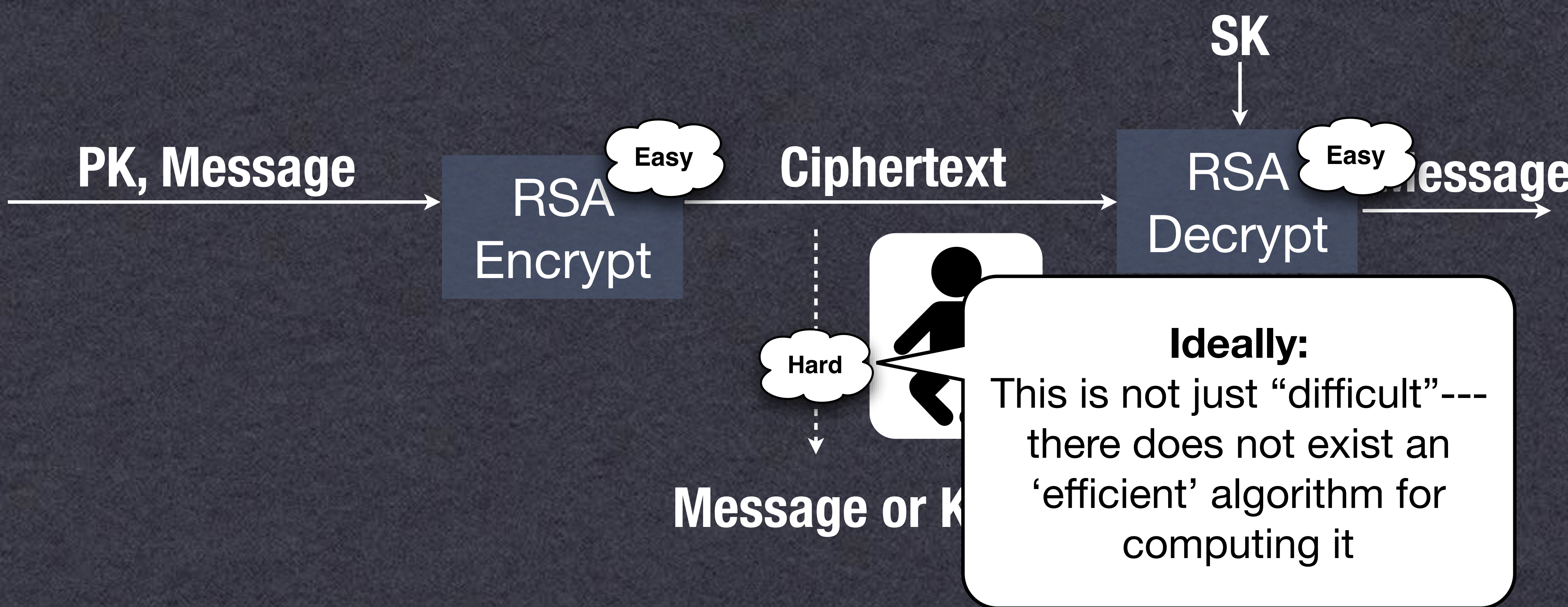
Computational Security

- AKA Complexity-Theoretic Security
  - RSA, AES/DES encryption, DSA, etc.
  - Security can be broken with enough effort
  - But the effort is enormous
- Proofs rely on some <u>hardness</u> assumption, eg:
  - RSA assumption
  - Discrete Logarithm
  - Factoring
  - Stronger assumptions: secure block ciphers (PRP)

Example



Example



SK
PK, Message
RSA Encrypt
Easy
Ciphertext
RSA Decrypt
Easy
essage
Hard
Message or K
Ideally:
This is not just “difficult”--- there does not exist an ‘efficient’ algorithm for computing it

Computational Security

- Problem & solution
  - Analyzing each new cipher is hard
  - Better: analysis of <u>a small number</u> of problems
  - No need to re-certify every new scheme
- Disadvantages:
  - We don't know if complexity-theoretic security is even possible!

One-Way Functions

- **Hypothesis:**
  - There are mathematical functions that are efficient to compute in one direction, but <u>cannot be efficiently computed</u> in the other
  - Theoreticians: “efficient” == polynomial time
- **Implication:**
  - If one-way functions exist then P != NP
  - One of the biggest open questions in theoretical Computer Science!

P = NP?

Clay Mathematics Institute

Dedicated to increasing and disseminating mathematical knowledge

HOME | ABOUT CMI | PROGRAMS | NEWS & EVENTS | AWARDS | SCHOLARS | PUBLICATIO

P vs NP Problem

Suppose that you are organizing housing accommodations for a group of four hundred university students. Space is limited and only one hundred of the students will receive places in the dormitory. To complicate matters, the Dean has provided you with a list of pairs of incompatible students, and requested that no pair from this list appear in your final choice. This is an example of what computer scientists call an NP-problem, since it is easy to check if a given choice of one hundred students proposed by a coworker is satisfactory (i.e., no pair taken from your coworker's list also appears on the list from the Dean's office), however the task of generating such a list from scratch seems to be so hard as to be completely impractical. Indeed, the total number of ways of choosing one hundred students from the four hundred applicants is greater than the number of atoms in the known universe! Thus no future civilization could ever hope to build a supercomputer capable of solving the problem by brute force; that is, by checking

- The Millennium Problems
- Official Problem Description — Stephen Cook
- Lecture by Vijaya Ramachandran at University of Texas (video)
- Minesweeper

Provable Security

P != NP

Trapdoor OWPs Exist

RSA is a Trapdoor OWP

Strength of assumption

What if P = NP

- Most theoreticians believe that P != NP
- If they're wrong, modern crypto's in trouble:
  - There may exist "polynomial-time" algorithms for inverting RSA, Elgamal, AES...**
  - We might have a hard time <u>finding</u> them
  - And "polynomial time" doesn't == super-fast

** Those schemes could be broken anyway...

Security Proofs

- Academic world: Late ‘70s, early ‘80s
    - Formal definitions of security
    - First schemes secure under mathematical assumptions
        - Proofs of concept
        - Some are “kind of” efficient
        - No ideas for efficient block ciphers, hash functions

Security Proofs

- Real world: Late ‘70s, early ‘80s, ‘90s
  - Nobody pays attention
  - <u>Heuristic</u> security
  - Provable schemes considered too expensive
  - Why?

Schnorr Signature

- **1990: Schnorr signature**
  - **Signature size = |p| + |q| (e.g., 1024 + 160)**
  - **Provably secure under Discrete Logarithm assumption in Random Oracle Model**

$$pk = p, q, g, g^x \ mod\ p \qquad sk = x$$

$$k \xleftarrow{\$} Z_q$$
$$c = H(g^k || M)$$
$$\sigma = (g^k \ mod\ p, xc + k \ mod\ q)$$

Schnorr Proof

- Board

DSA Signature

- 1991: Digital Signature Algorithm (US)
  - Signature size = |q| + |q| (e.g., 160 + 160)
  - <u>No security proof</u>

$$pk = p, q, g, g^x \ mod \ p \qquad sk = x$$

$$k \xleftarrow{\$} Z_q$$
$$c = H(M)$$
$$r = (g^k \ mod \ p) \ mod \ q$$
$$s = (k^{-1}(c + xr)) \ mod \ q$$
$$\sigma = (s, r)$$

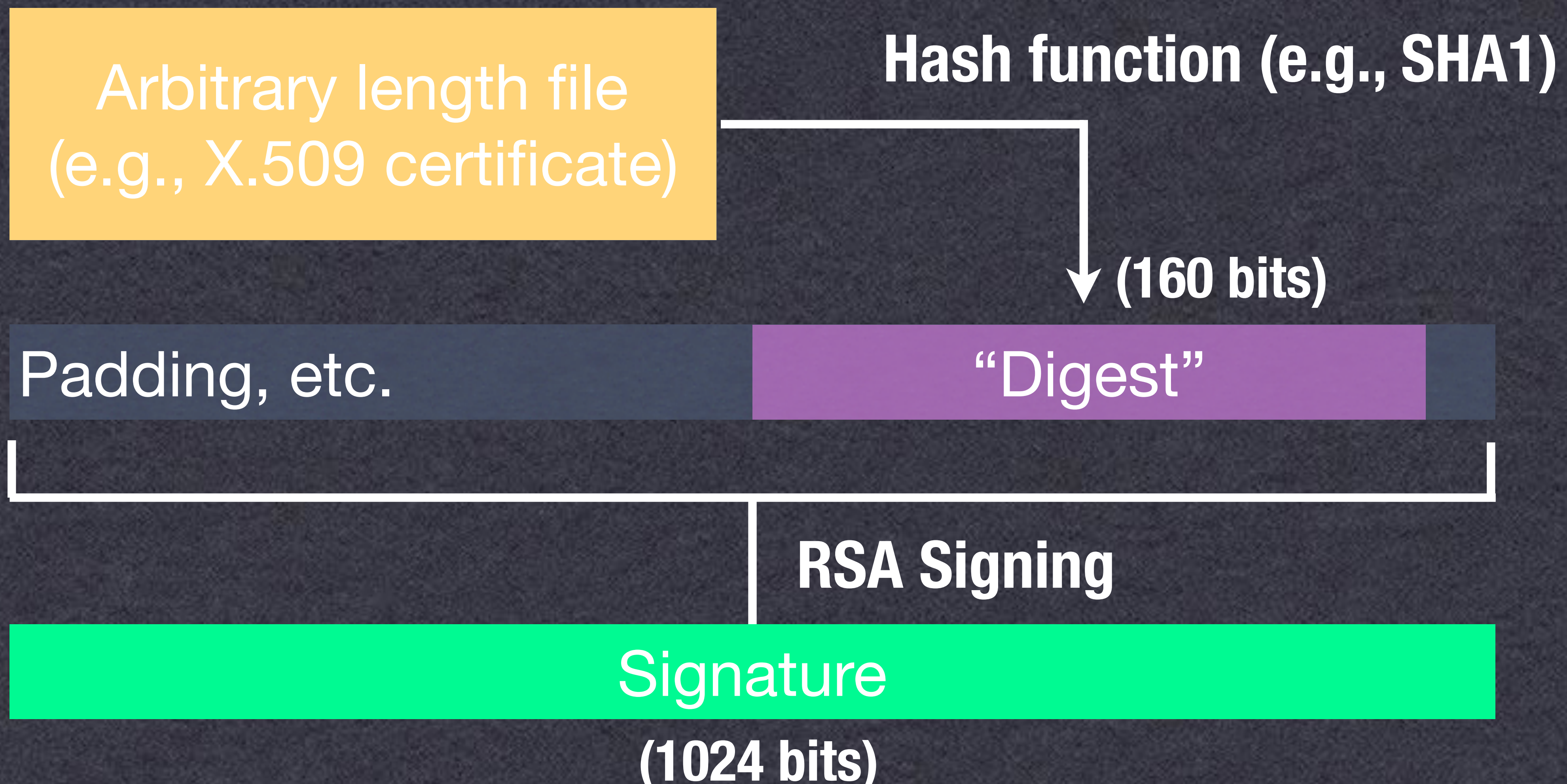
Hash Functions

- Convert variable-length string to small "tag"
  - Hash tables
  - Signatures
  - Software checksums
  - MAC functions (HMAC)
  - Encryption (OAEP)

Cryptographic

Signatures

- "Hash & Sign"
  - Allows us to sign arbitrary-sized files
  - Ex. RSA-PKCS signature:



Software Checksums

```
Debian GNU/Linux 4.0 alias etch
- -------------------------------

Source archives:

  http://security.debian.org/pool/updates/main/l/lighttpd/l...
    Size/MD5 checksum:     1108 d747ed7b2063ad6696064bf821c50a00
  http://security.debian.org/pool/updates/main/l/lighttpd/l...
    Size/MD5 checksum:    38244 c6de19903fcf9972a3db86af50c3dfb6
```

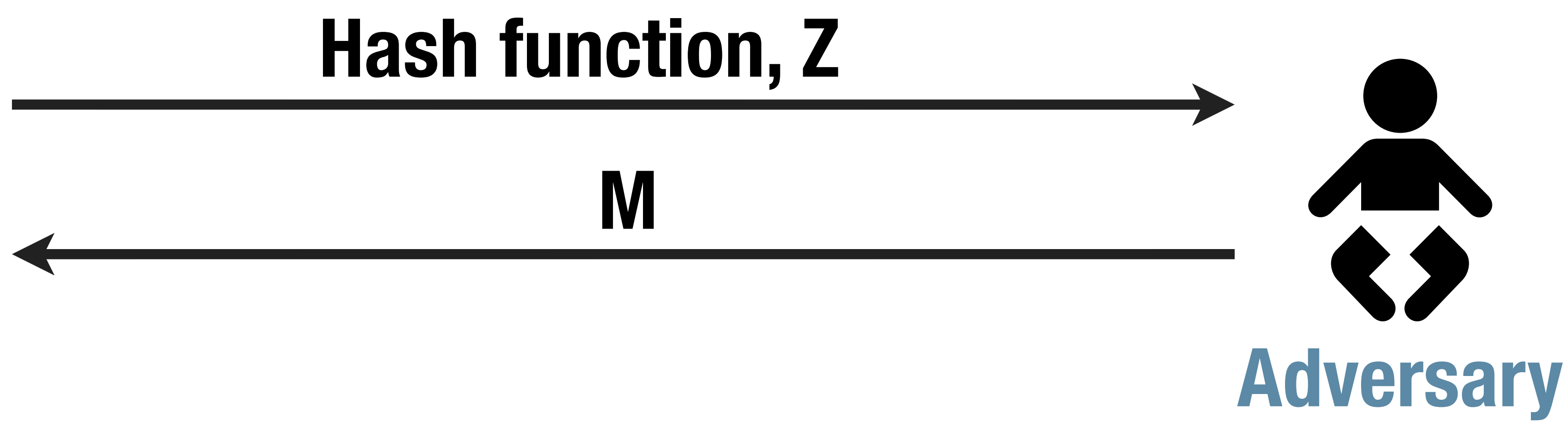
Cryptographic Hashing

- What, exactly, do we require?
- We have some guidelines:
  - Efficiency
  - Pre-image resistance
  - Collision Resistance
  - Second Pre-Image Resistance

Efficiency

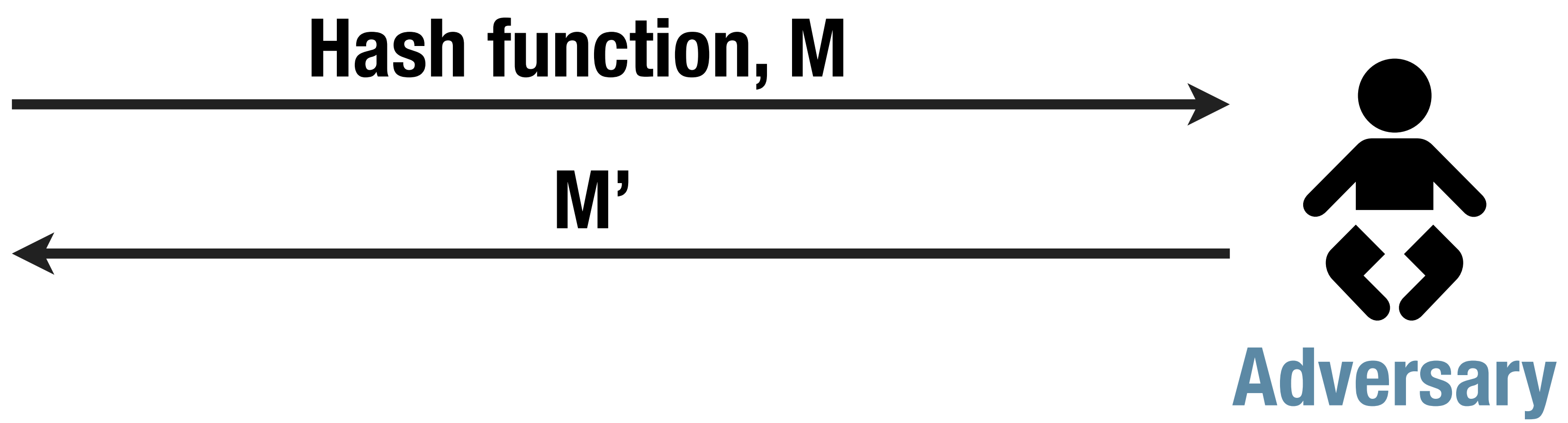
- Efficient to compute
- Algorithm is compact
- Theoretical definition:
  - computable in polynomial time (of input size)
  - this implies polynomial-size description

Pre-image Resistance



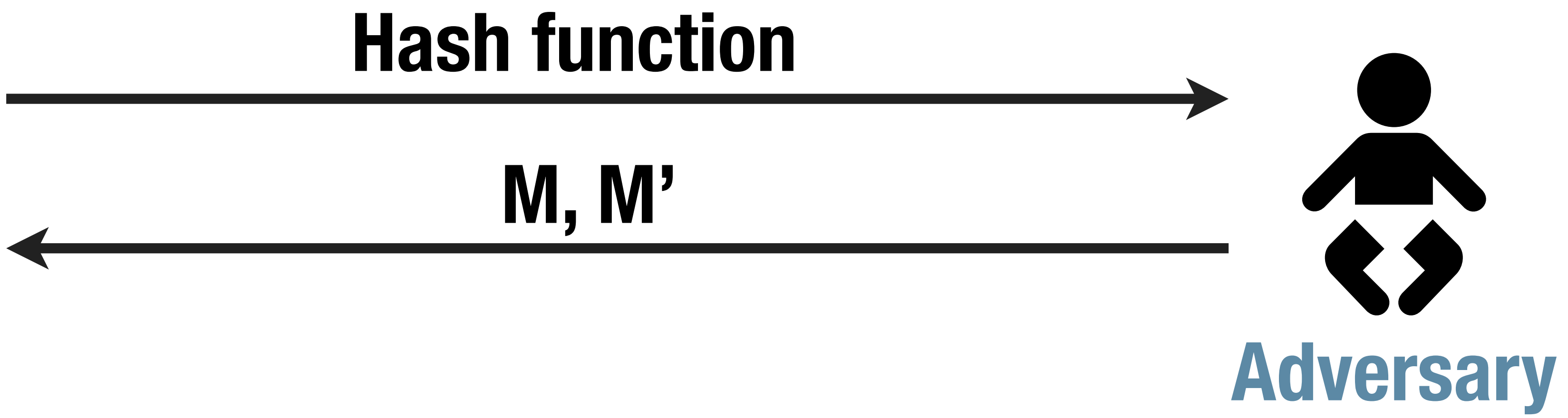
Adversary wins if Hash(M) = Z

2nd Pre-image Resistance



Adversary wins if Hash(M) = Hash(M’)

Collision Resistance



Adversary wins if Hash(M) = Hash(M’)

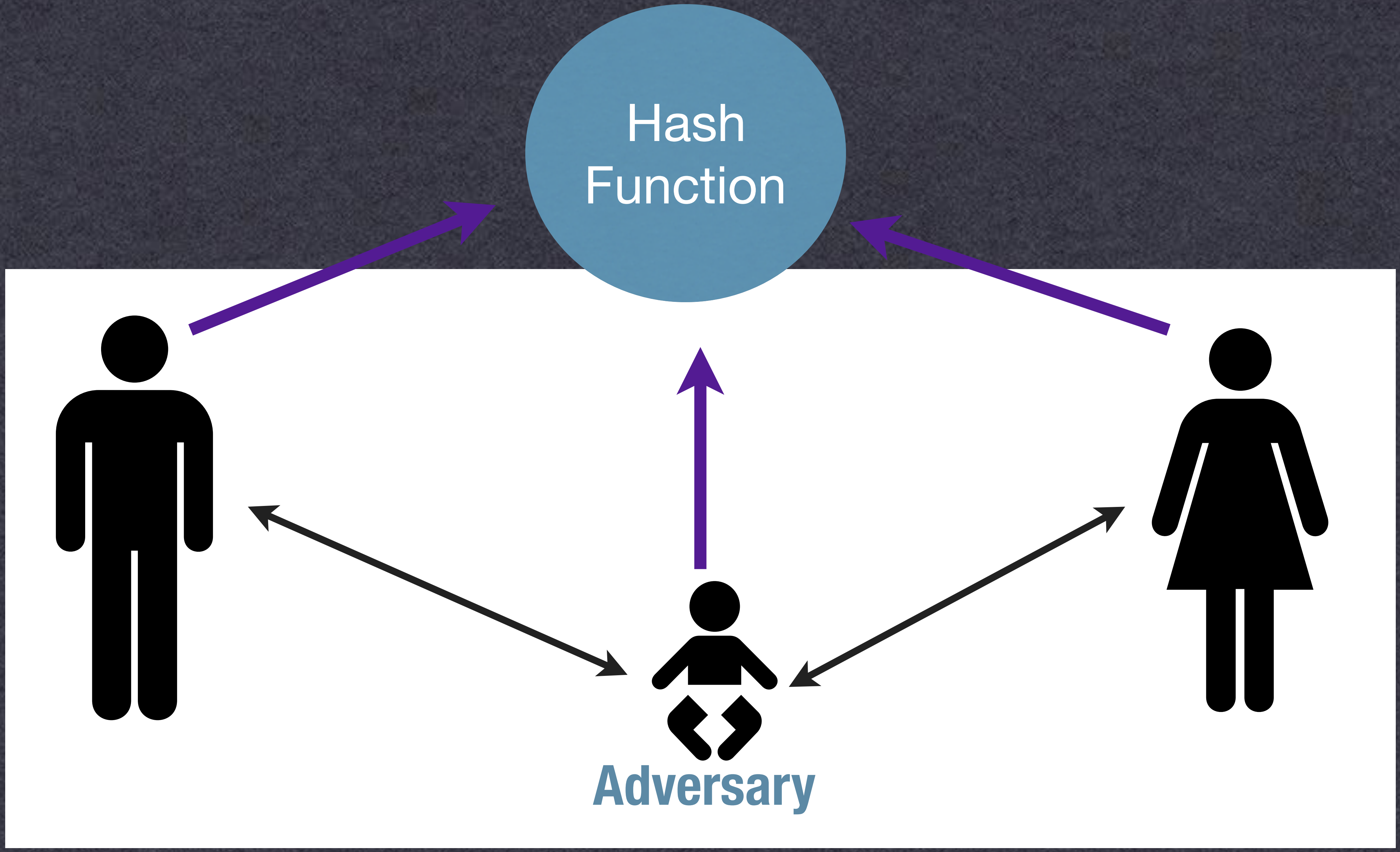
Ideal Hash Function

- What would a perfect hash function look like?
  - Outputs <u>completely</u> unrelated to inputs
  - E.g., a random function
- So...
  - H(M) leaks no special information about M
  - Collisions & 2nd preimages hard to find

Ideal Hash Function

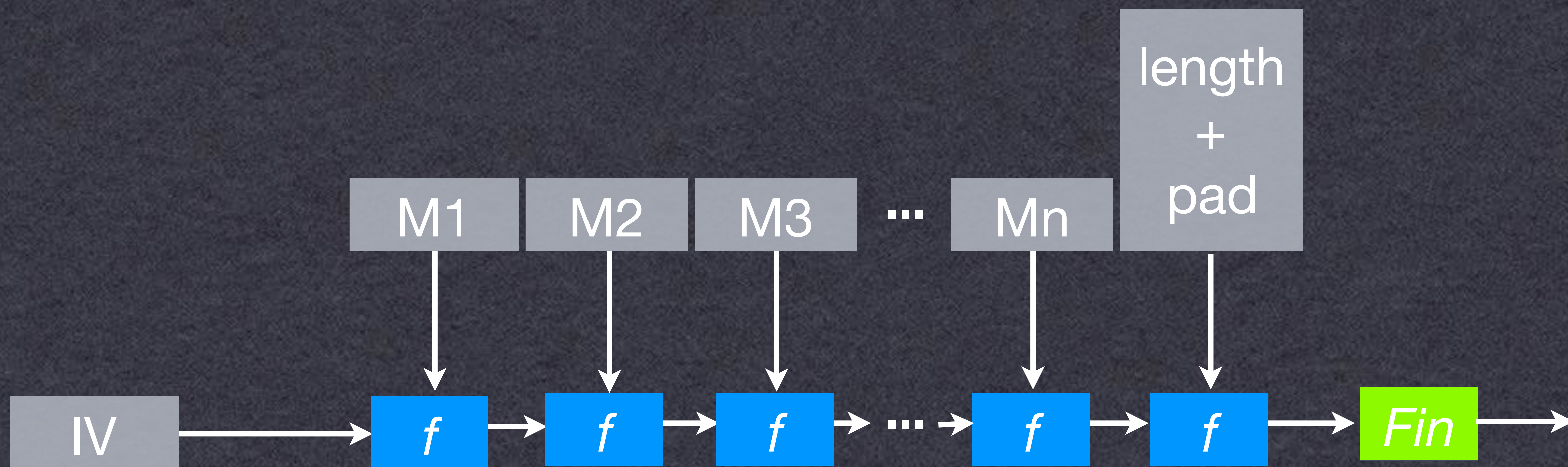
- Problem:
  - Random function description: exponential size
  - Takes exponential time to compute
  - But we want our algorithms to be fast & small (polynomial-time/space)
- Solution:
  - Don’t use ideal function at all (best)
  - Make a special exception so we can use it in our proofs (next best)

Random Oracles



Merkle-Damgård

- **Used in most standard hash functions**
  - (MDx, SHAx)

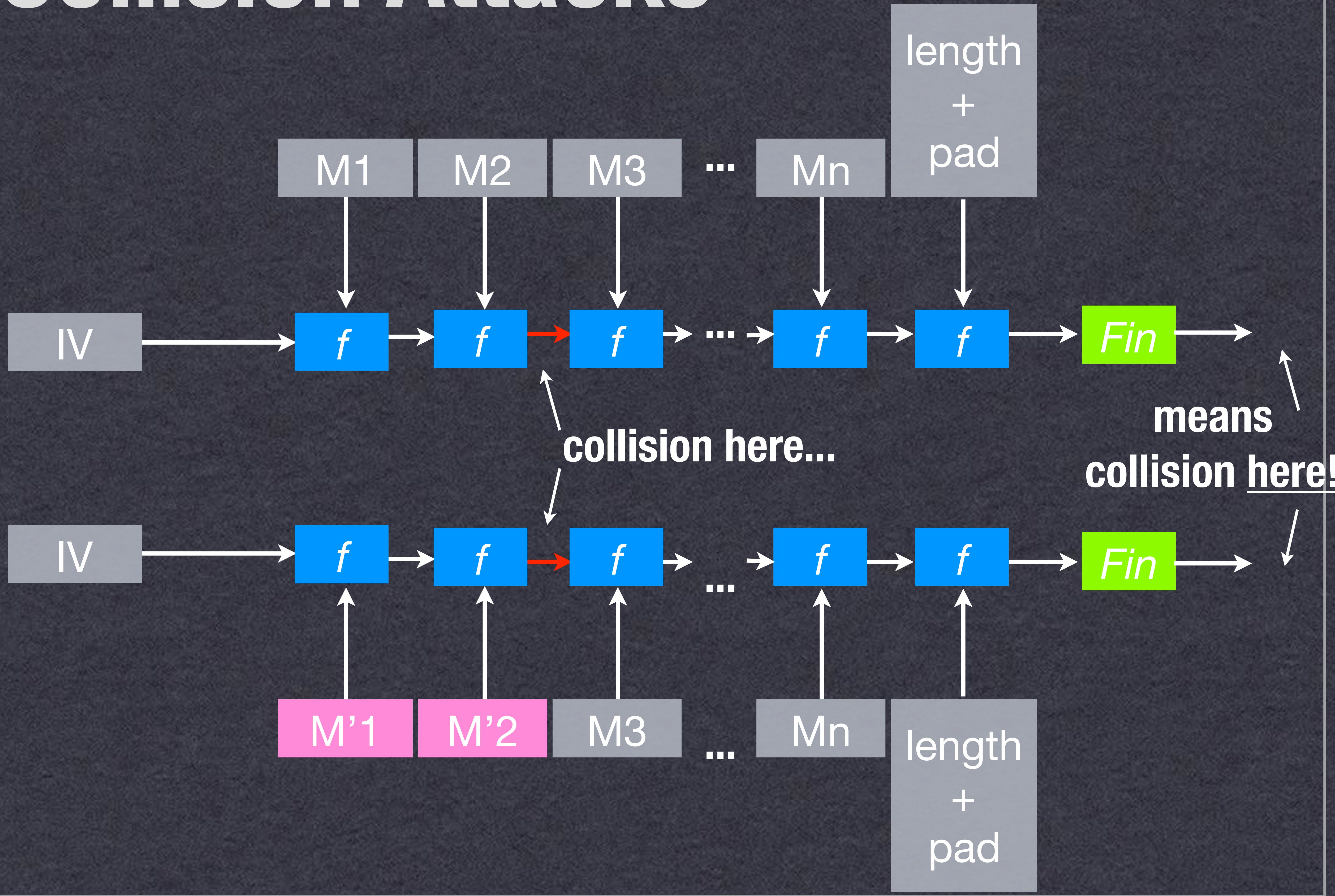


f = compression function (m x n-bit input)

Fin = (optional) finalization function

- Why Merkle-Damgard?
  - If f is collision-resistant, then H() is too (Crypto ’89)
  - If f is an ideal cipher (random function), then H() is an ideal hash function
  - But what if f is not collision-resistant?

Collision Attacks



“Textbook” RSA

Key Generation

Choose large primes: p, q

$$N = p \cdot q$$

$$\phi(N) = (p-1)(q-1)$$

Choose:

$$e \; : \; gcd\,(e, \phi(N)) = 1$$

$$d \; : \; ed \; mod \; \phi(N) = 1$$

Output:

$$pk = (e, N)$$

$$sk = d$$

Encryption

$$c = m^e \; mod \; N$$

Decryption

$$m = c^d \; mod \; N$$

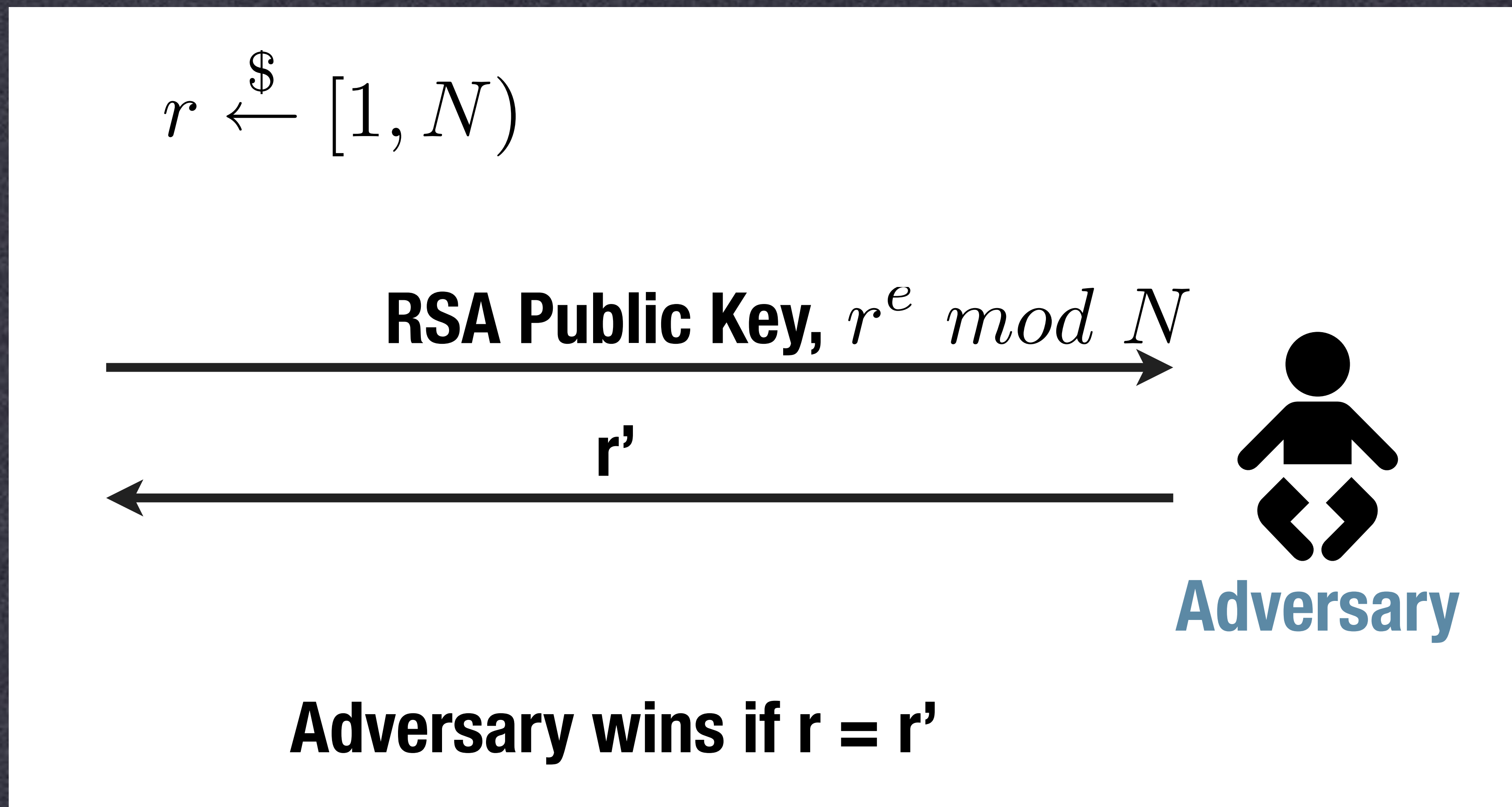
"Textbook RSA"

- In practice, we don't use Textbook RSA
  - Fully deterministic (not semantically secure)
  - <u>Malleable</u>

$$c' = c \cdot x^e \ mod \ N$$
$$c'^d = (m^e \cdot x^e)^d = m \cdot x \ mod \ N$$

  - Might be <u>partially</u> invertible
    - Coppersmith's attack: recover part of plaintext (when m and e are small)

RSA Assumption

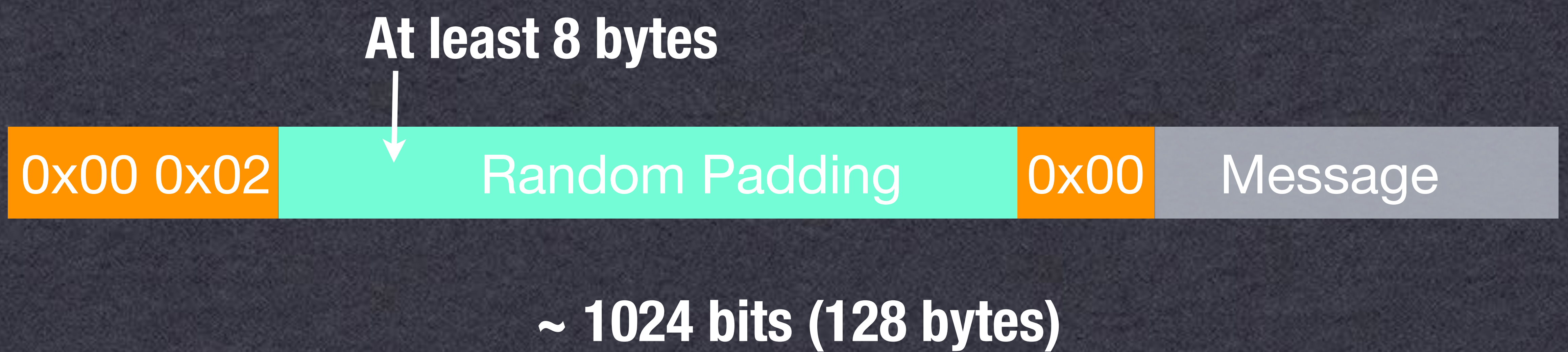


Implications:

1. Only holds if r is <u>random.</u>
2. Even if assumption is hard, might still be possible to partially decrypt.

RSA Padding

- **One solution (RSA PKCS #1 v1.5):**
  - Add “padding” to the message before encryption
  - Includes randomness
  - Defined structure to mitigate malleability



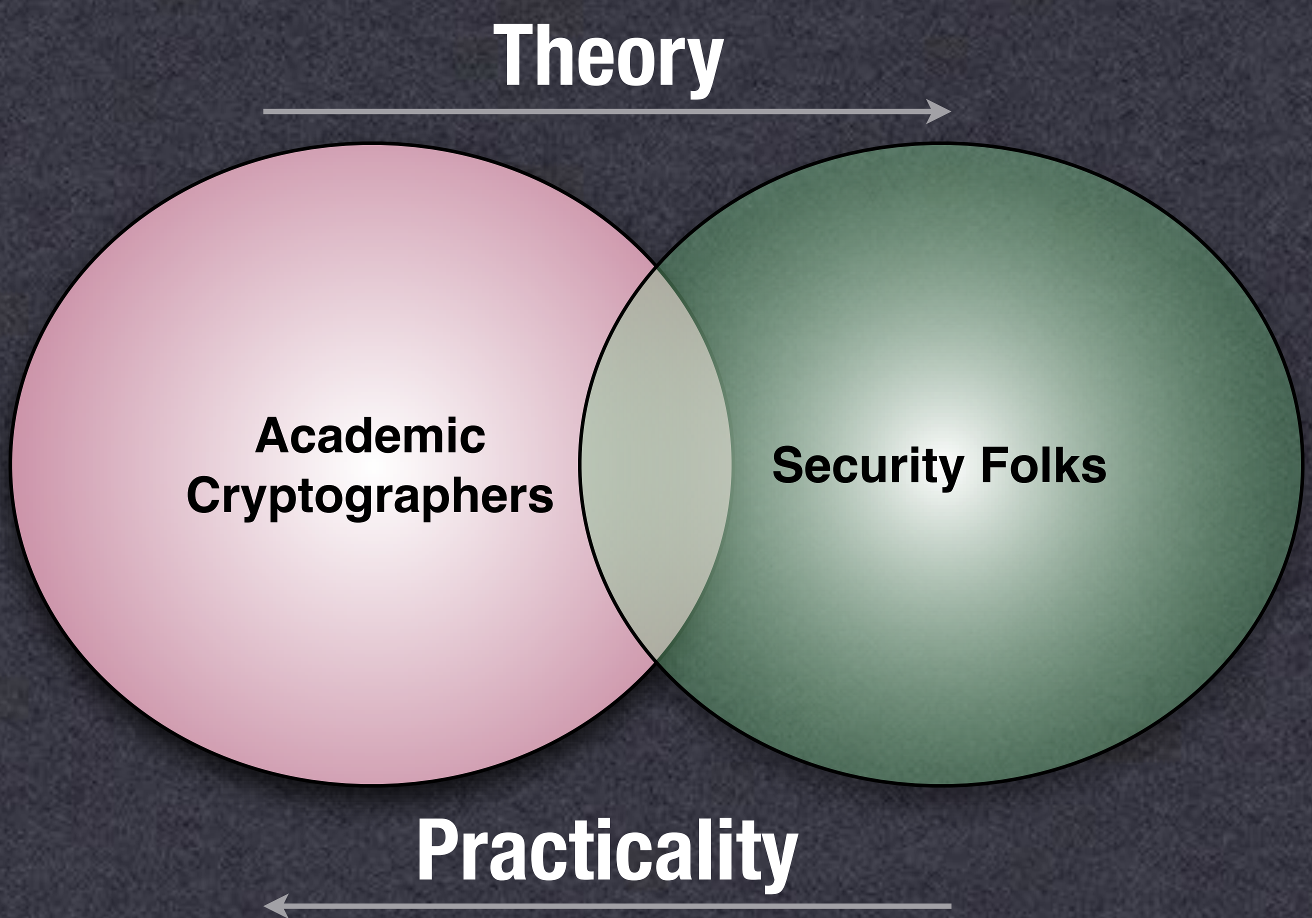
Key Diversification

This class

- **What happens when provable techniques are ignored?**

The Curious Case of PKCS#1 v1.5

Review


Theory
Academic Cryptographers
Security Folks
Practicality